GEOLOGICAL SURVEY OF CANADA

OPEN FILE 1334

MAGRAV2: An Interactive Magnetics and Gravity Modelling Program for IBM-Compatible Microcomputers

John Broome

August 1986

$30.00

MAGRAV2: An Interactive Magnetics and Gravity Modelling Program for Microcomputers *

John Broome

Lithospheric Geophysics Section

Lithosphere and Canadian Shield Division

Geological Survey of Canada

ABSTRACT

MAGRAV2 is an interactive program for modelling magnetic and gravity data that runs on IBM personal computers (IBM-PC), or compatibles. The program allows forward and inverse 2.5 dimensional modelling of gravity and magnetic anomalies from up to 10 bodies. Bodies are defined by their vertical cross-section and their strike extent.

The program is written for a microcomputer equipped with a high-resolution colour monitor in addition to a standard text monitor. The high-resolution colour monitor is used to display measured and calculated anomalies as well as the colour-coded body cross-sections while the text monitor displays only text information. Program control from either a graphics tablet or the keyboard is possible.

The software is written in Microsoft FORTRAN 77 with one 8088/8086 assembler subroutine for sound generation. Colour graphics are handled by the Multi-Halo graphics subroutine library. Multi-Halo is device-intelligent, allowing this software to be adapted to numerous different graphics peripherals. Other software for this workstation configuration to allow display and enhancement of geophysical imagery will also be made available in the future.

TABLE OF CONTENTS

APPENDICES :

1.0) INTRODUCTION

Magnetic and gravity data are often interpreted in a two stage process. The first stage involves qualitative analysis of the data displayed in map form, such as colour intensity maps, shaded relief images or contour maps. Anomaly trends are correlated to known geology and areas of interest are isolated. The second stage involves quantitative analysis such as forward and inverse modelling of data extracted from areas of interest. Forward magnetic modelling involves defining bodies with specified magnetic properties and calculating the theoretical anomaly that would be produced by the body. This calculated anomaly is compared to the measured anomaly and adjustments are made to the body shapes and magnetic parameters until a reasonable match is obtained between the measured and calculated anomalies.

Modelling can be done in 2, 2.5 or 3 dimensions. Two-dimensional modelling involves defining bodies in cross-section and assumes that the bodies have infinite strike extent. In 2.5 dimensional modelling, the bodies are still defined in cross-section but strike extent is variable. Anomalies from both 2 and 2.5 dimensional modelling are displayed in profile form. In three-dimensional modelling, where the geometry of the bodies is variable in three dimensions, the measured and calculated anomalies are displayed in map form. Early computer modelling efforts involved calculation of two-dimensional model anomalies by batch job submission to mainframe computers, analysis of the results, and resubmission of thejob with modified model parameters. This process was repeated until a satisfactory match was obtained between the measured and calculated anomalies. This batch type modelling gradually evolved to 2.5 and 3 dimensions.

The next major improvement was the development of interactive two-dimensional modelling programs for mainframe computers that utilized monochromatic graphics displays. One example of this type of program is MAGRAV (Haworth et al.,1980)(Wells,1979) which was written in FORTRAN 4 for CYBER mainframe computers and Tektronix storage-tube terminals at the Atlantic Geoscience Centre. The original MAGRAV used the two-dimensional modelling algorithms published by Talwani and Heirtzler (1964). This program was subsequently improved at the Geological survey of Canada (GSC) by P. McGrath (McGrath et al.,1983), who added inverse modelling capability and F. Lindia who added the end corrections to the two-dimensional magnetic modelling algorithms (Shuey et al.,1973) to make them 2.5-dimensional.

This program, MAGRAV2, is a new version of MAGRAV rewritten in Microsoft FORTRAN 77 to run on IBM personal computers or compatibles. MAGRAV2 incorporates the improvements made to the original MAGRAV by McGrath and Lindia as well as additional improvements added by the author.

To fully utilize MAGRAV2 additional hardware must be added to the basic microcomputer to improve its graphics and computational performance, as well as it's storage capacity. MAGRAV2 uses a high-resolution colour monitor to generate detailed colour graphics and an optional graphics tablet for cursor positioning and program control.

This open file includes a 360 kbyte IBM-format floppy disc containing source code, test model, and batch files to simplify compilation and linking of the program. After a discussion of the hardware required to use the program, the procedure to create an executable file and usage of the program will be described.

2.0) HARDWARE REQUIREMENTS

The hardware configuration described here was carefully selected to create a functional inexpensive geophysical workstation. Other software will be released in the future which requires this particular equipment configuration; therefore, this configuration is recommended to assure that your workstation will be compatible with this software. MAGRAV2 was written to operate most effectively on the complete system; however, provisions have been made for users who do not have all the equipment recommended for a complete workstation.

MAGRAV2 has three modes of operation to suit different hardware configurations :

Mode 1) Text display only
Mode 2) Colour graphics display with keyboard control
Mode 3) Colour graphics display with graphics tablet control.

Table 1 summarizes the hardware requirements to run MAGRAV2 in the different modes. Essential components for each mode are identified with an "E", recommended components with an "R" and optional components with an "O". Recommended components are those that are not essential for the particular MAGRAV2 mode but are essential for future software releases designed for the workstation.

2.1) MODE 1 REQUIREMENTS

MAGRAV2 operating in mode 1 will operate on any IBM-PC or compatible with 256 kbytes of memory and 2 floppy disc drives. The addition of an 8087 numeric processor chip is strongly recommended because it accelerates anomaly computation by a factor of approximately ten. Without an 8087 chip, a typical anomaly calculation for one body requires approximately 20 seconds for a 50 point profile; much too long for an interactive environment. With the 8087 chip the delay for anomaly calculation is less than 2 seconds. Although not essential, a hard disc unit

Component	Mode: 1	2	3
IBM-PC or compatible .... -IBM-PC/XT or AT is optional -Running MS-DOS or PC-DOS	E	E	E
256 kbyte memory .... -640 kbyte recommended	E	E	E
2 - 360 kbyte floppy discs .... -only 1 floppy disc is required if a hard disc or 1.2 Mbyte floppy disc is available.	E	E	E
Serial port (for digitizer tablet)..................	R	R	E
Parallel port (for printer)	O	O	O
8087 numeric processor chip .... -Strongly recommended!	R	R	R
Text monitor and display board	E	E	E
High resolution colour graphics board .... -Supplied software is designed for: the Number Nine Computer Corp., "Revolution" board, 512 x 512 x 8 bit, interlaced	R	E	E
High resolution RGB Colour monitor	R	E	E
Graphics tablet .... -The software is setup for a Houston Instruments Hipad digitizer Model DT-11	R	R	E
Printer .... -a standard printer is useful for program listings and model dumps	O	O	O
Hard disc .... -optional but recommended for ease of compilation,linking and program operation	O	O	O

TABLE 1 : Hardware required for different modes of operation. "E" indicates essential, "R" indicates recommended as this device is essential for other workstation software, and "O" indicates optional.

is recommended to simplify program compilation and linking and to speed program operation. In Mode 1, all information, including observed and calculated anomaly profiles, are typed out in numerical form rather than being presented graphically as in modes 2 and 3. Program operation is controlled by 38 different command options which are explained by an on-line help function. Although all program functions are available in mode 1, the absence of graphic display of anomaly profiles and body cross-sections makes the modelling process slower and more difficult.

2.2) MODE 2 REQUIREMENTS

Mode 2 operation requires the addition of a high-resolution colour monitor and driver board to display anomaly profiles and body cross-sections. The program is controlled by the keyboard using the same 38 different command options used for mode 1. Body cross-sections and anomaly profiles are drawn colour-coded for easy recognition. The graphic display resolution of the standard IBM colour graphics adapter is inadequate for this application. Therefore, a high resolution graphics board was added to the computer. The particular graphics board used in this system is the Number Nine Computer Corp.'s Revolution board. This board produces an interlaced 512 by 512 pixel display with 256 simultaneously displayable colours out of a palette of over 16 million colours. MAGRAV2 itself does not require 256 colours but other software designed for this workstation requires this capability. Other colour graphics boards can be used with the program because all of the graphics are controlled by the device-intelligent Multi-halo graphics library. Considerations for adapting the program for different hardware are outlined in Appendix E.

2.3) MODE 3 REQUIREMENTS

Mode 3 operation requires the addition of a digitizer tablet to the computer. This particular system uses a Houston Instruments Hipad digitizer (model DT-11). In mode 3, program option selection and body point movement are controlled from the graphics tablet using the graphics tablet cursor. Program control from the graphics tablet is achieved by placing a template over the graphics tablet that identifies areas on the graphics tablet corresponding to different program options. As in modes 1 and 2 the text monitor is used to display prompts, informative listings and error messages. Mode 3 operation is the most interactive modelling environment and the recommended mode of operation.

3.0) COMPILING AND LINKING MAGRAV2

3.1) SOFTWARE REQUIREMENTS FOR COMPILING AND LINKING

Before MAGRAV2 can be used , an executable file must be generated by compiling the FORTRAN source code files and linking them with the FORTRAN, Halo, and MAGRAV2 libraries. To produce an executable MAGRAV2 file, two commercial software products are required :

1) The Microsoft FORTRAN 77 compiler (version 3.20 or higher Required for all modes)
2) The Multi-Halo graphics subroutine library with Microsoft FORTRAN 77 support (version 2.26 was used For graphics in modes 2 and 3)

The Microsoft FORTRAN 77 compiler is required for all modes of operation to compile the five FORTRAN source code files. The source code is broken into five files because there is too much code to be compiled as one module by the compiler. The batch file "mfcomp.bat" can be used to compile the five files as outlined in Appendix D. The contents of the five source code files; "magrav2.for", "msl.for", "ms2.for", "ms3.for", and "ms4.for" are listed in Appendix C. All of the subroutines are required for mode 3 of operation; however, operation in modes 1 and 2 does not require all the subroutines. A brief description of the purpose of each subroutine, which modes of operation required, and the source code file in which it is found are included in Appendix B.

To generate an executable file for a system with no digitizer tablet, files not required for modes 1 and 2 can be edited out of the source files. Calls to the deleted subroutines must also be deleted from the main program and other subroutines, or error messages will occur during linking.

Subroutine SOUND is an 8088/8086 assembler routine, found in "magrav2.lib" which is used to generate sound to accompany program prompts and error messages. It is not essential to program operation and can be left out if all calls are edited out of the FORTRAN source code.

3.2) THE MULTI-HALO GRAPHICS LIBRARY

The Multi-Halo graphics subroutine library is used to produce the graphic display used in modes 2 and 3 of operation. Multi-halo is a device-intelligent system for handling graphics on microcomputers produced by Media Cybernetics of Silver Spring, Maryland. Device intelligence allows software to be used with different hardware with minimal changes. Device driver files are provided for many common microcomputer graphics boards, printers and positioning devices such as digitizers and mice. These drivers are installed at run time to allow software to be used in different hardware environments. Modifications that may be required for different graphics boards and digitizers are discussed in Appendix E. Mode 1 operation does not use any Multi-Halo subroutines, so an executable MAGRAV2 file can be generated for mode 1 operation, by commenting out any Multi-halo subroutine calls in the FORTRAN source code files before compiling and linking.

The object files produced during compilation and assembly must be linked to each other and to the FORTRAN and Multi-halo libraries to produce the executable MAGRAV2 file. The batch file "mlink.bat" can be used to link the files as described in Appendix D.

4.0) SETTING UP TO RUN THE PROGRAM

4.1) FILES USED BY MAGRAV2

Once a "magrav2.exe" file has been produced you are ready to model. A number of files are required by MAGRAV2 and the program generates others. A list of these files and their purpose follows :

1) Magrav2.exe :
MAGRAV2 executable file you generate by compiling and linking the FORTRAN source code files provided.

2) Halo.dev :
Device driver file used by the halo graphics. For the 512x484x8 number nine graphics board, this file is Halo file "halonine.dev" renamed "halo.dev. This file and others are provided with the Halo graphics package. "Halo.dev" must be located on the default drive.

3) Logo.pic :
This file is optional . It is a Halo format image file produced by the Halo "gwrite" command. If the file is found on the default drive, the stored image will be displayed on the colour monitor when MAGRAV2 is executed. You can generate your own "logo.pic" file if you have the 'Dr. Halo' image editing program.

4) Models file :
The models file is generated by the MAGRAV2 and contains models stored by using the "write" option in MAGRAV2. You may name this file whatever you wish. This file allows the user to save models and read them back later for further modelling or inspection. Sample models files "mtest.mod" and "gtest.mod" are provided.

5) Recovery file :
This is a scratch file generated on the default drive by MAGRAV2 containing information used by the "recover" option to allow modelling steps to be undone. This file is named "magrav.rec" in the program and takes up approximately 200 kb of disc space.

6) Init.bat :
This file is used to initialize the serial port on the IBM-PC for the Houston instruments Hipad digitizer for use with the program."Init" must be run before MAGRAV2.

7) Halohipi.com :
This file contains the device driver file for the Houston instruments digitizer tablet. It is provided with the Halo graphics package and used by "init.bat".

4.2) SETTING UP WITH A 1.2 MBYTE FLOPPY OR HARD DISC

If a hard disc or a 1.2 Mbyte floppy disc is included in the system, then files "magrav2.exe", "init.bat", "halohipi.com", "halo.dev" and the models file can all be placed on one drive.

4.3) SETTING UP WITH TWO 360 KBYTE FLOPPY DISCS

If two 360 kbyte floppy discs are available all the files used by MAGRAV2 will not fit on one drive. In this case, files "halo.dev", and the models file should be placed on the default drive and a disc containing "magrav2.exe", "init.bat", and "halohipi.com" should be placed in the other disc drive.

4.4) OPTIONAL USE OF A RAM DISC TO IMPROVE PERFORMANCE

After each significant model change, MAGRAV2 writes a block of data to file "magrav.rec" for use by option "reco" for undoing changes. If this write is to a floppy disc, program operation can be slowed considerably. If 640 kbytes or more memory is available, the program can be speeded up by 'installing' a 360 kbyte RAM disc in memory and making it the default drive. Files "halo.dev" and the models file should be placed on the default drive and MAGRAV2 should be executed from another drive. "Magrav.rec" will now be created on the RAM disc and writes to "magrav.rec" will be performed much more quickly. A commercial software package such as "Superdrive" by AST research or the MS-DOS 3.1 configuration option "vdisc.sys" can be used to create the RAM disc.

4.4) INITIALIZING THE DIGITIZER

If MAGRAV2 is to operate with digitizer tablet control (mode 3), the system must first be initialized for the appropriate digitizer by running batch file "init.bat". This sets up the serial communications port on the computer for the particular locator device used. "Init.bat" executes Halo file "halohipi.com" to initialize the port for the Houston Instruments Hipad digitizer.

5.0) USING MAGRAV2

MAGRAV2 is started by entering the command "magrav2". The program will first ask for the name of the models file. A test models file, "mtest.mod", provided on the disc, should be on the default drive; so enter "mtest.mod". The program will then ask if graphics are to be enabled. To use the program in mode 1 enter "n" and for modes 2 and 3 enter "y". The program will then prompt you with "Enter option:".

5.1) PROGRAM OPTIONS AND HELP

MAGRAV2 is controlled by 38 four character program command options. Any of the options can be called at any time; however, a logical sequence must be called. Obviously, the option which moves a body cannot be used if no bodies have been defined. If an attempt is made to select an option that cannot be used, a message will be generated to identify the error. The correct starting order of option calls is given by option "help" together with a menu of the possibilities. Within "help", information describing each of the options can be obtained by entering the 4-character name. Pressing the "return" key, returns you to the main program. Two sequences of option calls can be used to get started, depending on whether the model is being generated for the first time or an existing model is to be read in from the models file.

5.2) TESTING THE PROGRAM

The models file is used to store model information so the user does not have to enter the observed and body data each time he wants to work on the model. To test the program, read in the model from models file "mtest.mod". To read in a model one must first determine the names of models stored in the models file. Option "tnam" will list the names of the models stored in the models file. Model "test" should be listed. Set the current model name to "test" using option "name" and then use option "read" to read model "test" into the program. Since "test" is a magnetics model, the program will switch from the default gravity mode to magnetics mode. Model "test" contains a 50 point measured magnetic data profile, magnetic field parameters, and 4 source bodies with their magnetic properties. To calculate the anomaly due to the bodies call option "anom". The program will type "Calculating anomaly for body n" where n is 1 to 4 as the anomalies for each of the bodies is calculated and then the "Enter option:' prompt will return. The calculated anomaly can now be printed out with option "tano". The body point coordinates and magnetic parameters can be printed out with option "tbod". If graphics are enabled (mode 2) the observed and calculated data can be plotted on the graphics monitor along with the cross-sections of the 4 bodies using option "draw".

5.3) USING THE DIGITIZER

If a digitizer tablet is included in the system, mode 3 of operation can be used. Mode 3 of operation allows most program options to be selected from a Houston Instruments digitizer tablet. The template, shown in figure 1, is placed over the working surface of the digitizer which identifies areas which correspond to different program options. If a Houston instruments digitizer is used, ensure that the digitizer tablet is placed in stream mode. To enter mode 3 of operation, graphics must be enabled and option "tabl" called. Program options can now be selected by placing the digitizer cursor over the appropriate command square on the digitizer and pressing the cursor button. Additional instruction are then printed on the text monitor. The

MAGRAV2

ENTER BODY	MOVE BODY	ENTER PARAM	MOVE POINT AUTO	MOVE POINT MANUAL	RECOVER	TEXT MODE	ANOMALY	DRAW	REDRAW
DELETE BODY	BODY ANOMALY		DELETE POINT	INSERT POINT			OPTIMISE CONTRAST	SKETCH	SCALE
TYPE BODY	TYPE PARAM	TYPE OBSE	TYPE ANOM	MAGN	GRAV	DIFF ON/OFF	OFFSET	SET ZOOM	MANUAL SCALE

Figure 1 : Program control from the digitizer tablet (mode 3) is obtained by placing the digitizer cursor over the template square representing the desired option and pressing the cursor button. This template is designed for use with a Houston Instruments digitizer and is shown reduced in size. The width and height of the black border on the template should be 25.4 cm.

digitizer tablet provides a much easier and faster method for changing body points than using the keyboard.

6.0) MODELLING YOUR OWN DATA

To model your own data, execute MAGRAV2 and then select the name of the models file you wish to use. If the program does not find the file you have selected it will ask you if you wish to create a new models file with that name . If you respond "n" the program will again prompt you for the models file name. After the models file is selected and the graphics mode is set, the first option selected should be "grav" or "magn" depending on the type of modelling to be done. If magnetics mode is selected, the program will ask for the orientation of the profile data, declination and dip of the geomagnetic field in the profile area, and whether a depth offset is desired for body points. The depth offset is useful for aeromagnetic data modelling since it can be set equal to the survey flight elevation to allow body point depths to indicate the depth below the earth's surface. If gravity mode is selected, the program will ask for the depth offset only. The next step is to enter the observed data values.

6.1) ENTERING OBSERVED DATA

The observed data is entered using option "eobs". The program will first ask whether data is to be read from a file or entered manually from the keyboard. Data entry from a file is provided as a link between the modelling program and digital sources of profile data such as field magnetometers with internal storage or airborne profile data. The format for profile files is described in Appendix F.

If manual entry is selected, the program will ask for the "x" profile offset. The "x" profile offset is a constant added to observed data positions which is useful when profiles longer than the current maximum of 100 points are to be modeled. The long profile can be modelled in two or more sections with the "x" offset set so that the "x" coordinate of the last point of the first section equals that of the first point of the next section. The program next asks for the observed data sampling interval and the number of profile readings to be entered. The current maximum profile length of 100 points could be increased by simply re-dimensioning the appropriate arrays in the program. The appropriate number of profile point numbers and data values are now entered. If graphics are enabled, the observed data will be automatically scaled and plotted in green on the colour monitor.

6.2) ENTERING BODIES

Once the observed data has been entered, the next step is to define the body cross-sections by entering body points. At this stage, an understanding of potential field interpretation and the geology of the area becomes important. Potential field

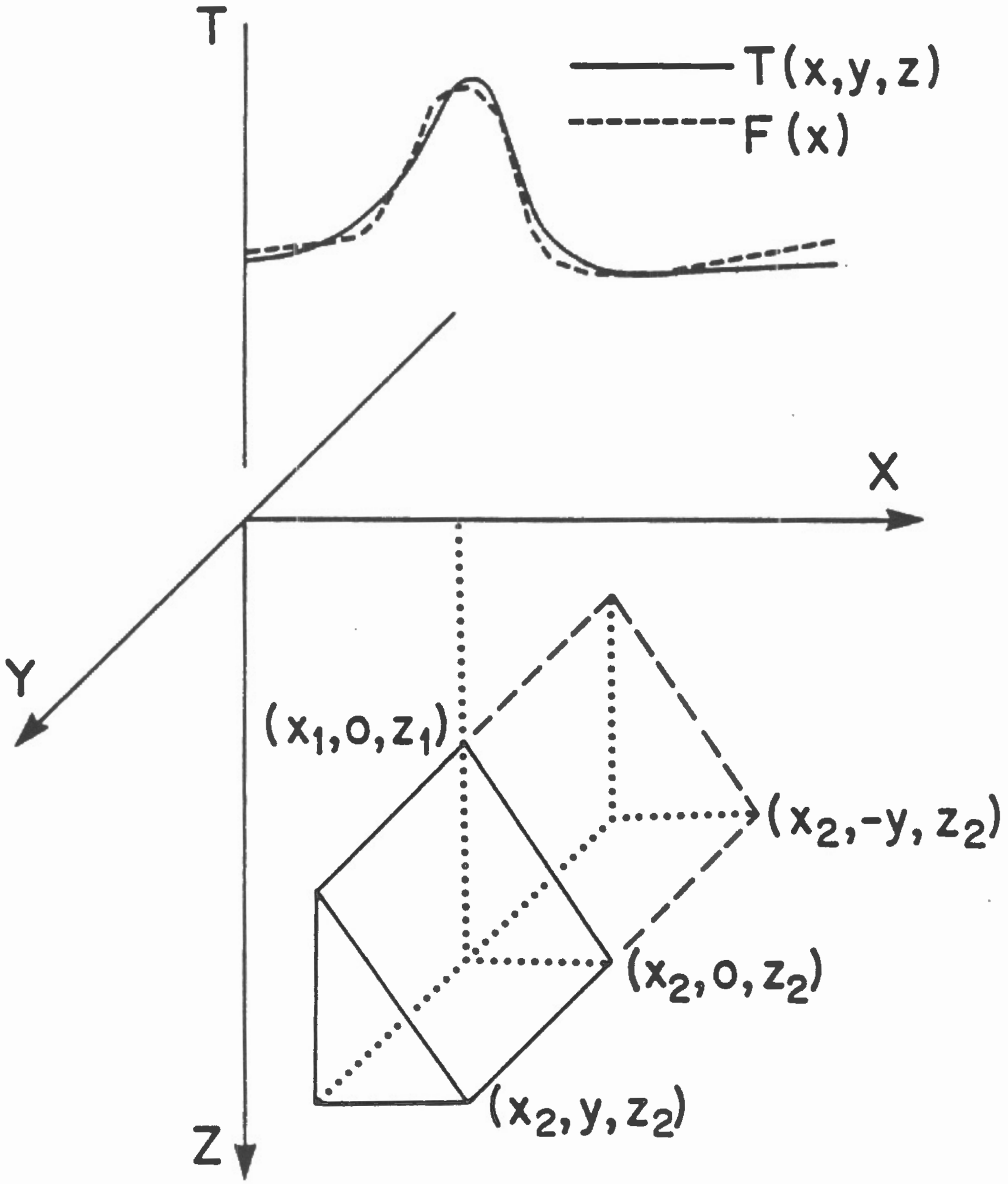

Figure 2 : This diagram shows the geometry used by MAGRAV2 to define bodies. The screen display shows only the x-z plane through points (x1,0,z1) and (x2,0,z2). The half-strike length distance, or strike extent, used by the 2.5 dimensional modelling algorithm is equal to y.

interpretation is complicated because of the ambiguity problem which results in an infinite number of combinations of body geometries and magnetic properties that will produce an anomaly which matches an observed anomaly. Geological constraints such as the measured magnetic susceptibilities on the profile and knowledge of the structure and contact locations between zones with contrasting susceptibility assist in obtaining a model which is realistic. Body points can be entered from the keyboard using option "ebod" (modes 1,2,3) or more easily from the digitizer tablet (mode 3). Up to 20 points can be entered per body and points must be entered in clockwise order or anomalies will not be calculated correctly. Bodies will be plotted, in colour, on the graphics monitor if graphics are enabled. Figure 2 shows how bodies are defined and displayed by the program. Up to 10 bodies can be created. Each body has a unique number and colour. The body colour is identified in the print-out produced by option "tbod".

As you enter body points they are compared to existing body points. If the position of the new point is very close to an existing point, the new point position is set equal to the old. The separation between points must be greater than the size of the small box in the lower left corner of the screen for the new point position to be retained. The next step in defining your model is to enter the body parameters.

6.3) ENTERING BODY PARAMETERS

Body parameters are entered using option "epar" or by selecting enter parameter on the digitizer tablet(mode 3). The first parameter the program will ask for is the body strike extent. The strike extent is used by the 2.5 dimensional algorithms to calculate the anomalies. The strike extent entered is the distance from the central cross-section of the body to each end of the body. The length of the body is therefore, twice the strike extent with the anomaly calculated over the center cross-section. Next, the program will ask for the minimum,actual and maximum magnetization or density contrast depending on whether you are in gravity or magnetics mode. The minimum and maximum values are limits used by the automatic contrast setting option, "cont", to constrain the contrasts to a limited range. This range would be set from measurements of density or susceptibility of rock samples along the profile or by looking up representative ranges for rock types identified along the profile.

The magnetic susceptibility of a rock is a measurement of the degree to which the rock can be magnetized. Since rock magnetic susceptibility is the usual rock property measured in the field, the relationship between magnetic susceptibility and magnetization is important for use of the program. The magnetization is the magnetic moment per unit volume and is related to magnetic susceptibility as follows :

$k = M / H$ where in cgs units;

k = Magnetic susceptibility (dimensionless)
M = intensity of induced magnetization (emu/cc)
H = intensity of the geomagnetic field (oersted)
(1 oersted = 100000 gamma)

Magnetization is expressed in cgs units of 0.00001 emu/cc in the program. Conversion of susceptibility from cgs to SI units is achieved by multiplying by 12.57.

If you are in magnetics mode the program will also ask for the declination and dip of the body magnetization. For most magnetic modelling, magnetization is induced by the geomagnetic field and is therefore in the same direction as the geomagnetic field. If remanent magnetization is present, the direction entered should be the direction of the vector sum of the remanent and induced magnetization vectors. The actual magnetization value entered in the program will be the amplitude of the vector sum.

6.4) CALCULATING AND DISPLAYING THE ANOMALY

Once body points,body parameters, and observed data have been entered, the model anomaly may be calculated using option "anom". The program will ask you to enter the body number for which the anomaly is to be plotted. Entering a specific body number will result in the anomaly from that body being plotted colour coded to the body cross-section plot. Entering "0" will result in the composite anomaly from all defined bodies being plotted in white. The anomaly can also be listed using option "tano", or if graphics are enabled, drawn on the colour monitor using option "draw" or "sket". If changes to body points or parameters are made, the anomalies will be automatically recalculated when options "draw" or "sket" are called and the composite anomaly will be plotted in white.

The observed and calculated anomalies are automatically scaled to fill the plot window on the graphics screen the first time it is drawn. If the observed data contains a constant background it can be removed for plotting purposes by calling option "offs". Offset and model changes may require rescaling of the anomaly plot window. This rescaling can be done automatically with option "asca" or manually with option "msca".The zero level in the anomaly window is indicated by a gray line. When option "asca" is called, both old and new plot limits are printed on the text screen.

7.0) OPTIMIZING THE MODEL

Once the body and anomaly data have been entered into the program, other options may be called to modify different model parameters. The model can be changed both manually and automatically by the program. When optimizing the model, care must be taken to ensure that the model remains relevant to the known geological constraints. Ambiguities inherent in magnetic interpretation result in an infinite number of models that will fit the observed anomaly. Manual optimization of the model is recommended in the early stages of developing the model since the interpreter can keep the model within reasonable geological constraints. Automatic methods are useful when the model is thought to be close to being correct, to make the final adjustments.

7.1) MANUAL OPTIMIZATION

After the initial model has been created, adjustments to model parameters are almost always required to improve the match between the observed data and calculated anomaly. Manual changes to body cross-sections can be made from the keyboard with options: "mpoi", "dpoi", and "ipoi" or from the digitizer tablet in mode (3). Manual changes to magnetization or density contrasts can be made with option "epar". Information on the other options that can be used for changing the model can be obtained using the "help" option.

7.2) AUTOMATIC OPTIMIZATION

The program also allows the user to automatically move body points using option "maut" and optimize body density or magnetization parameters using option "cont". Both automatic optimization methods generate a best least-squares fit between the calculated and observed data. The automatic optimizations work properly only if the model produces a calculated anomaly reasonably close to the observed data. If the body point or body contrast is poorly constrained, the automatic point movement may produce unreasonable results due to the mathematical accuracy limitations of the program.

Option "cont" will vary the density or magnetization contrasts of the body within the minimum and maximum limits entered in option "epar". This allows realistic limits to be placed on body parameters to conform to the known geology. If option "cont" is used for optimizing a magnetic model where the body magnetization has been set to a different direction to account for remanent magnetization, the model will be optimized by changing the magnitude of the resultant magnetization vector. As explained previously, the resultant is the vector sum of an induced component determined by the susceptibility of the body and a fixed remanent component. Since the remanent component is fixed, the induced component should be optimized to fit the model rather than the resultant which indicates the total magnetization. For this reason, automatic contrast optimization is not recommended where a remanent magnetization component is included in the model.

8.0) OTHER PROGRAM FUNCTIONS

8.1) RECOVERY

During the optimization process, changes are often made which are undesirable. Option "reco" allows the user to undo the last 20 changes of the program. After each significant change to the model the model is saved in file "magrav.rec" and each call to option "reco" undoes one change.

8.2) ZOOM

The screen resolution can be a limitation to the modelling process for complex or very detailed models. For this reason a zoom function has been implemented which allows a portion of the screen display to be blown up to fill the screen. The area to be zoomed can be defined using option "zoom" or using the set zoom command on the digitizer tablet. Once the zoom area has been defined, the user can switch back and forth between full profile display by calling option "draw" and zoom area display by calling option "sket". All digitizer command options work in both the "sket" and "draw" display modes. Using "zoom" to look at a extremely small area will take some time and may cause the program to fail.

Zooming out to increase the display beyond the ends of the profile can also be accomplished using option "msca" to set the zoom area manually from the keyboard. This is a useful feature which allows bodies to be extended beyond the profile ends to eliminate edge effects.

8.3) SAVING THE MODEL

If model changes are made, the revised model must be written to the models file, before exiting from the program, using option "writ"; otherwise, changes will be lost. The model name can be left unchanged, in which case the previous model will be overwritten, or the model name can be changed to create a new model in the models file. A maximum of 20 models can be stored in a models file.

8.4) SIMULTANEOUS GRAVITY AND MAGNETIC MODELLING

MAGRAV2 allows simultaneous modelling of both gravity and magnetics for a given model. To use this feature enter either a gravity or magnetic model as described and then call either option "grav" or "magn" to change to the other mode. After the modelling mode is set, enter observed data and body parameters for that mode. Once both gravity and magnetic observed data and body parameters have been entered, the current modelling mode can be switched back and forth by calling options "magn" and "grav". Modelling of both gravity and magnetic data allows the model to be constrained more completely than using one mode alone.

9.0) CONVERTING MAGRAV2 FOR VERTICAL GRADIENT MODELLING

MAGRAV2 can be easily modified for vertical gradient modelling by making some changes to subroutine MAG. The modifications use the depth offset to calculate the magnetic anomaly at two different heights one meter apart. The difference between these anomalies is an approximation to the vertical gradient of the field. The observed vertical gradient data should be entered in units of gammas/metre to match the calculated anomaly. Appendix G contains a modified version of subroutine MAG called MAGVG. This method of calculating the vertical gradient doubles the amount of computation required therefore anomaly calculation takes twice as long.

10.0) MODIFICATIONS AND DISTRIBUTION

Many individuals have expressed interest in MAGRAV2 and other programs used by this microcomputer based workstation . By releasing this program into the public domain, is is hoped that the program will receive widespread use. This use will probably lead to modifications, improvements and corrections to errors that may exist in the program. The author would appreciate if a description of any significant modifications or corrections to the program could be sent to me so that they can be incorporated in later versions of the program. Please distribute only unmodified versions of the program.

11.0) DISCLAIMER

This program is provided on an "as is" basis. Neither The Geological Survey of Canada nor any of its staff members are liable for any errors in the program or any problems associated with use of the program.

12.0 REFERENCES

Haworth,R,T. and Wells,I.
1980; 'Interactive computer graphics method for the combined interpretation of gravity and magnetic data'; Marine Geophysical Researches, No. 4, p. 277-290, D. Reidel Publishing Co.

McGrath,P.H., Henderson,J.B., and Lindia,F.M.
1983: 'Interpretation of a gravity profile over a contact zone between an Archean granodiorite and the Yellowknife Supergroup using an interactive computer program with partial automatic optimization'; in Current Research, Part B, Geological Survey of Canada., Paper 83-1B, p. 189-194.

Shuey,R.T. and Pasquale,A.S.
1973; 'End corrections in magnetic profile interpretation'; Geophysics, v. 38, no. 3, p. 507-512.

Talwani,N., and Heirtzler,J.R.
1964, 'Computation of magnetic anomalies caused by two-dimensional bodies of arbitrary shape'; in G.A. Parks ed.), Computers in the Mineral Industry, School of Earth Sciences, Stanford University.

Wells,I.
1979; 'MAGRAV users guide: A computer program to create two-dimensional gravity and/or magnetic models',; Bedford Institute of Oceanography Computing Services Technical Services Memorandum No. 85, Geological Survey of Canada.

APPENDIX A

Files supplied on the disc :

-Microsoft FORTRAN source files:

1 magrav2.for
2 ms1.for
3 ms2.for
4 ms3.for
5 ms4.for
6 magrav.cmn

-Assembler routines file:

7 sound.asm

-Libraries:

8 magrav2.lib

-Batch files:

9 mfcomp.bat
10 mlink.bat

-Models file:

11 mtest.mod
12 gtest.mod

APPENDIX B

Subroutine description :

1) Name..modes requiring it.... location of the source code
2) What it does.
3) Other routine calls
(Argument descriptions are given in the source code)

NOTE: Subroutine calls not found in this list are calls to the"Halo" graphics subroutine library.

In alphabetical order :

ASCALE M: 2,3 MS2.FOR

This subroutine automatically scales the anomaly plots to fit the screen window.

CALCAN M: 1,2,3 MS3.FOR

This subroutine calls the appropriate gravity or magnetic anomaly calculation subroutine to calculate the anomalies from any bodies with new or changed parameters or point positions.

CHECK M: 3 MS3.FOR

This subroutine compares body point positions to existing body points. If the new point position is within the distance specified in variables "xdis" and "zdis" from an existing point a flag is set. This check removes the need for absolutely accurate cursor positioning.

DEGCOS M: 1,2,3 MS1.FOR

This function calculates the cosine of an angle input in degrees.

DEGSIN M: 1,2,3 MS1.FOR

This function calculates the sine of an angle input in "degree".

DELE M: 1,2,3 MS2.FOR

This subroutine prompts the user, in text mode, for body numbers and point numbers for point deletion.

DELETE M: 1,2,3 MS3.FOR

This subroutine deletes points in bodies.

DELTAG M: 1,2,3 MS1.FOR

This function is used by subroutine "GRAVC" when calculating gravity anomalies.

DFS001 M: 1,2,3 MS1.FOR

This subroutine is used to optimize point positions and gravity or magnetic contrast values. A more detailed description of the program and parameters is given in the source listing.

DFS002 M: 1,2,3 MS1.FOR

This subroutine is used by DFS001 to find the best fit of the calculated gravity or magnetic anomaly for a particular degree of freedom. A more detailed description of the subroutine and parameters is given in the source listing.

EOBSE M: 1,2,3 MS2.FOR

This subroutine prompts the user to enter observed data offset, sample spacing, number of readings in the profile, and observed data values from the keyboard

GRAVC M: 1,2,3 MS3.FOR

This subroutine calculates the gravity anomaly for one body.

GRINIT M: 2,3 MS2.FOR

This subroutine initializes the Halo graphics, loads the graphics device driver, and sets the colours for the screen display.

HELP M: 1,2,3 MS2.FOR

This subroutine prints out informative text messages describing the different command options.

INSE M: 1,2,3 MS2.FOR

This subroutine prompts the user (in text mode) for the body number, point number, and point coordinates for a point to be inserted an existing body.

INSERT M: 1,2,3 MS3.FOR

This subroutine inserts new body points

MAG M: 1,2,3 MS3.FOR

This subroutine calculates the magnetic anomaly for one specified body.

PARAM M: 1,2,3 MS3.FOR

This subroutine prompts the user to enter body gravity/magnetic parameters from the keyboard.

POIN M: 1,2,3 MS2.FOR

This subroutine prompts the user (in text mode) for body numbers, point numbers and new point coordinates for point movement.

READF M: 1,2,3 MS2.FOR

This subroutine is used to read a model from the models file with the current name.

REAN M: 1,2,3 MS3.FOR

This subroutine clears arrays "calc" and "ianom" to zero for initialization purposes.

RECO M: 1,2,3 MS2.FOR

This subroutine is used to read the most recent record from the recovery file into the common block. This allows the user to go back to previous steps by "undoing" changes.

ROBSE M: 1,2,3 MS2.FOR

This subroutine allows the user to read observed profile data from an ASCII file.

SAV M: 1,2,3 MS2.FOR

This subroutine is used to write the common block to the recovery file after every significant change so that previous steps can be recovered with option "RECO".

SOUND M: 1,2,3 (not essential) SOUND.ASM

This 8086/8088 assembler subroutine makes sounds of different frequency and duration to accompany error messages and prompts. This subroutine can be removed without affecting the utility of the program.

TSCA M: 1,2,3 MS3.FOR

This subroutine types plot scaling parameters, and other informative information about the current program status.

TYPANO M: 1,2,3 MS3.FOR

This subroutine types the composite or individual calculated anomalies on the text monitor.

TYPAR M: 1,2,3 MS3.FOR

This subroutine types the gravity or magnetic contrast parameters for all bodies on the text monitor.

TYPBOD M: 1,2,3 MS3.FOR

This subroutine types body points and parameters for selected bodies or all bodies.

TYPOBS M: 1,2,3 MS2.FOR

This subroutine is used to print outi the observed data on the text monitor in text mode.

WHAT M: 1,2,3 MS2.FOR

This subroutine is used to interpret text mode commands. A character string read in from the keyboard is compared to a set of commands and an value used in computed "go to" statements is returned.

WRITEF M: 1,2,3 MS2.FOR

This subroutine is used to write a model with the current name to the models file.

APPENDIX C

```
----------  :**************************************************************

c                           --------------------
c                          |      MAGRAV2       |
c                           --------------------
c
c           INTERACTIVE GRAVITY AND MAGNETICS MODELLING
c
c            Revision 1.1 ; edited June 19 / 86
c
c ** NOTE: Source code for this program is in 5 files :
c          magrav2.for, ms1.for, ms2.for, ms3.for, ms4.for
c          File: magrav.cmn is included in all source files.
c
c          Libraries for compilation :
c          Microsoft fortran libraries, Halo graphics library(2.26)
c          and assembler object file :sound.
c
c *********************************************************************
c
c  This program is an improved version of MAGRAV, the history of
c      which follows:
c
c  Author: I. Wells, Computing Services ,Bedford institute of
c  Oceanography(BIO). for Dr. Haworth, Atlantic Geoscience Centre.
c
c Reference :
c  Wells, I.(1979) MAGRAV - A computer program to create two
c   dimensional gravity and/or magnetic models,
c   Computer Science Centre OPEN FILE 597.
c
c  Magnetics modified by Franca M Lindia, August 1982.
c  Modifications based on 2.5-D magnetic equations of
c  Shuey and Pasquale (1973).
c
c  Gravity equations modified by Peter McGrath, December 1982,
c  for 2.5 D bodies. See Rasmussen and Pederson (1979)
c  Geophys. Prosp., 27, 749-760.
c  Other minor modifications to the program were also made.
c
c  Modified by Peter McGrath, December 1982, to permit
c  automatic adjustment of body magnetization (density)
c  contrasts and point movement using a non-linear
c  least squares algorithm published by Powell (1965) in
c  The Computer Journal, 7,p 303.
c
c  Rewritten in Microsoft FORTRAN 77 for use on an IBM-PC with
c  a raster graphics display driven by the HALO graphic system
c  by : John Broome, Lithospheric Geophysics Section,
c  Lithosphere and Canadian Shield Division, Geological Survey
c  of Canada, June 1986.
c -------------------------------------------------------------------
c  Input/output:
c  =============
```

```
c
c Input : Input from keyboard or control from digitizer pad.
c Recovery file : "magrav.rec" used to allow you to go back
c          to a previous step.
c Models file : Contains named model data so you can terminate
c       a modeling session and then continue later.
c ---------------------------------------------------------------------
c  Variables in common blocks (Magrav.cmn) :
c  =========================================
c
c  ANOMAX(NTYPES): Maximum of anomaly window for plotting observed
c     and calculated anomalies.
c  ANOMIN(NTYPES): Minimum of anomaly window for plotting observed
c     and calculated anomalies.
c  BDEC(MAXBOD): Magnetization declination of each body. (mag only)
c  BDIP(MAXBOD): Magnetization dip for each body. (Mag only)
c  BDY(MAXBOD): Half-strike length of each body in km.
c  CALC(MAXBOD,MAXOBS): Anomaly calculated for each body,
c     at each observed point,CALC(MAXBOD+1,MAXOBS) stores the
c     combined anomaly from all bodies.
c  DEC: Declination of field. (Mag only)
c       0 - 360. Degrees clockwise from North.
c  DIFMAX(NTYPES): Maximum of anomaly plotting window for difference
c        plots.
c  DIFMIN(NTYPES): Minimum of anomaly plotting window for difference
c        plots.
c  DIP: Dip of the field. (Mag only)
c       0 - 360. Degrees
c  E(22): Array used by DFS001
c  F(MAXOBS): Array used by DFS001 to store the calculated anomaly
c          values.
c  IANOM(MAXBOD): 0 Anomaly has not been calculated for this body
c                -1 Anomaly has been calculated for this body
c                   and is stored in calc
c  IB(4): Used by subroutine AMOVE to store body numbers for cases
c         where a point is found in more than 1 body.
c  IBR: Used in automatic optimization, set=1 for point movement
c          set=0 for contrast optimization.
c  IDIFF: -1 for observed and calculated plot
c          1 for difference between observed and calculated
c  IRECOV(NBACK): set to "1" if that rec. no. has been saved
c  ISCOPE: 1 All graphics in effect
c          2 All graphics suppressed, for text-only terminals
c  ISCR: Logical unit number for recovery file
c  ITYPE: 1 for gravity mode modelling.
c         2 for magnetics mode modelling.
c  IX(LNGIX): character array containing current model names
c  JF: Digitizer tablet button status(4=pushed)
c  JX:     "        "       X   coordinate
c  JZ:     "        "       Z        "
c  KALK: Set=1 if anything changes that requires anomaly recalculation
c  KOMMNT(8): 8A10 comments about model
c  LNGIX: Max number of models that can be stored
c  MAXBOD: Maximum allowable number of bodies
c  MAXCAL: Maximum allowable number of bodies + 1.
```

```
c  MAXNPT: Maximum allowable number of points per body
c  MAXOBS: Maximum allowable number of observations
c  MODE: Set=1 - for full profile plotting(draw mode)
c        Set=2 - for plotting of only the "zoomed" window
c  MODEL: Unit number of model file
c  NAME: Name of current model entered with "name" option
c  NBACK; Current record number in recovery file.
c  NBODS: Number of bodies in current model
c  NFIELD(NTYPES): number of profile points for each modelling mode
c  NMOD: Current number of defined models in models file.
c  NP(4): Used by subroutine AMOVE to store point numbers in cases
c         where a point to be moved occurs in more than 1 body.
c  NPTS(MAXBOD): Number of points currently in each body
c  NTYPES: Number of model types this program will handle. (Mag,Grav)
c  OBS(MAXOBS,NTYPES): Observed values for each mode and profile pt.
c  OFFSET(NTYPES): Offset added to observed for plotting only
c  PI: Pi constant
c  RHOMAG(MAXBOD,ITYPE) Density(ITYPE=1), or Magnetization(ITYPE=2)
c        contrast for each body.
c  RMMAX(MAXBOD,NTYPES): Maximum value allowed for body contrast.
c  RMMIN(MAXBOD,NTYPES): Minimum value allowed for body contrast.
c  SKXMAX: Current right side sketch mode plotting limit
c  SKXMIN: Current left side sketch mode plotting limit.
c  SKZMAX: Current maximum depth for sketch mode plotting.
c  SKZMIN: Current minimum depth for sketch mode plotting.
c  SPACE(NTYPES): Spacing between field points
c  W(1275): Scratch work array used by DFS001, etc.
c  X(MAXNPT,MAXBOD): X Coordinates for each point in each body.
c  XC: Current X cursor position scaled to the current window.
c  XDIS: Distance in scaled units in X direction for resolution
c        on pinpointing for cursor input of bodies
c  XLEN(NTYPES): Length of profil in Km.
c  XLPL: Current right X plotting window limit
c  XMAX: Position in km of the last profile point.
c  XOFFS(NTYPES): X Offset of first profile reading in km.
c  XPOS(MAXNPT): X coordinate of each observed reading on profil
c  XTON: X to N angle. Orientation of X axis. (Mag only)
c      measured clockwise from North in degrees.
c  XUPL: Current left plotting window limit
c  XX(11): Array used by DFS001 to store variables being modified
c         for the best fit.
c  Z(MAXNPT,MAXBOD): Z Coordinates of each body point
c        Note - (0,0) is not an acceptible body point
c  ZC: Current Z cursor position scaled to the current window.
c  ZCON(ITYPE): A constant added to Z coord of all bodies
c  ZDIS: Allowable distance between the cursor and body point
c        positions in the Z direction.
c  ZLPL: Current bottom plotting window limit
c  ZMAX: Maximum distance from surface(0.) to bottom of screen
c  ZUPL: Current  top plotting window limit
c
c ********************************************************************
c
      program magrav2
c
```

```
      integer*2 digini
      character*10 iblank
      character*20 fil,mfil
      character*4 ians
      logical*2 fex
c
$include:'magrav.cmn'
$nofloatcalls
c
      data iblank/'          '/
      write(*,299)
299   format(1h0,'        MAGRAV2  1.1',/,
     +1h ,'        ------------',//,
     +1h ,' Gravity and magnetics modelling program',/,)
c
c -------------------------------------------------------
c Open and initialize models and recovery files
c
      lngix = 20
      iscr = 9
      model = 1
      nmod = 0
      do 200 i = 1 , lngix
        ix(i) = 0
200   continue
c
225   write(*,310)
310   format(1h0,'Enter name of models file : ',\)
      call sound(20,200)
      read(*,'(a)') mfil
      inquire(file=mfil,exist=fex)
      if(fex.eqv..false.) then
        write(*,*)'File ',mfil,' not found.'
        write(*,'(a\)')' Open a new model file(y/n): '
        call sound (20,200)
        read(*,'(a)') ans
        if(ans.eq.'n'.or.ans.eq.'N') go to 225
        open(model,file=mfil,status='new',access='direct',
     +  form='unformatted',recl = 3120)
        write(*,*)'New models file ',mfil,' opened'
      else
        open(model,file=mfil,status='old',access='direct',
     +  form='unformatted',recl=3120)
320     nmodpl = nmod + 1
        read(model,rec=nmodpl,end=330) moddat
        nmod = nmod + 1
        if(nmod.gt.lngix)go to 330
        ix(nmod) = name
        go to 320
330     write(*,*) nmod,' models read from file ',mfil
      endif
c
      open(iscr,file='magrav.rec',status='new',access='direct',
     +form='unformatted',recl=8320)
c
```

```
220   idiff = -1
      iscope = 2
      nback = 0
      itype = -9999
      kalk = 1
      nbods = 0
      maxnpt = 19
      maxbod = 10
      maxobs = 100
      maxcal = maxbod + 1
      ntypes = 2
      zmax = -1.
      skxmax = 0
      skxmin = 0.
      skzmin = 0
      skzmin = 0.
      xlpl = 0.
      xupl = 0.
      zlpl = 0.
      zupl = 0.
      xton = 0.
      xdis = 0.
      zdis = 0.
      mode = 1
      dec = 0.
      dip = 90.
      zcon(1) = .001
      zcon(2) = zcon(1)
c
      do 250 j = 1, ntypes
        offset(j) = 0.
        xoffs(j) = 0
        xlen(j) = 0
        anomax(j) = 100.
        anomin(j) = -100.
        difmin(j) = -50.
        difmax(j) = 50.
        nfield(j) = 0
        space(j) = 1.
        do 255 i = 1, maxobs
          obs(j,i) = 0.
255     continue
250   continue
c
      call rean
c
      do 260 i = 1 , maxbod
        npts(i) = 0
        rhomag(i,1) = 0.
        rhomag(i,2) = 0.
        rmmin(i,1) = 0.
        rmmin(i,2) = 0.
        rmmax(i,1) = 0.
        rmmax(i,2) = 0.
        bdec(i) = 0.
```

```
          bdip(i) = 0.
          bdy(i) = 100000
260     continue
c
        do 270 i = 1, 8
          kommnt(i) = iblank
270     continue
        name = iblank
c
        call grinit (iscope)
c
        write(*,'(/a/)')' Select option "HELP" to start'
        go to 400
c
c Branch to chosen option from value returned by "what"
c =======================================================
c
350     call sav
c
400     write(*,'(/a\)')' Enter option : '
        call sound(20,200)
        call what(iwhat)
        if(iwhat.le.0) then
499       write(*,*) 'pardon ?'
          goto 400
        endif
c
        if(itype.lt.0) then
          if(iwhat.eq.25.or.iwhat.eq.26.or.iwhat.eq.8)goto 500
          if(iwhat.eq.35.or.iwhat.eq.30.or.iwhat.eq.19)goto 500
          if(iwhat.eq.15.or.iwhat.eq.16.or.iwhat.eq.27)goto 500
          write(*,*)'The modelling mode must be selected with'
          write(*,*)'options "GRAV"ity or "MAGN"etics, or a '
          write(*,*)'model must be "READ" in before this option'
          write(*,*)'can be called.'
          go to 400
        endif
500     goto(1000, 2000, 3000, 4000, 5000, 6000, 7000, 8000,
     +      9000, 10000, 11000, 12000, 13000, 14000, 15000,
     +     16000, 17000, 18000, 19000, 20000, 21000, 22000,
     +     23000, 24000, 25000, 26000, 27000, 28000, 29000,
     +     30000, 31000, 32000, 33000, 34000, 35000, 36000,
     +     37000, 38000)
     +     iwhat
c
c <EOBS> Enter or read in observed data
c ------------------------------------
1000    if(itype.eq.1) write(*,*) 'You are in  GRAVITY MODE'
        if(itype.eq.2) write(*,*) 'You are in MAGNETICS MODE'
1014    write(*,'(a\)')' Is ths the correct mode (y/n) : '
        read(*,'(a)',err=99000) ians
        if(ians.eq.'y'.or.ians.eq.'Y') then
          write(*,*)' 1 - Read observed data from a profile file'
          write(*,*)' 2 - Enter observed data manually'
          write(*,'(/a\)')' Select type of data input(1 or 2): '
```

```
          read(*,*,err=99000) intype
          if(intype.lt.1.or.intype.gt.2) go to 99000
          if(intype.eq.1)then
            call robse(ierr)
            if(ierr.eq.0) then
              go to 350
            else
              go to 400
            endif
          else
            call eobse
            go to 350
          endif
        else
          if(itype.eq.1)write(*,*)'Select option "MAGN"etics'
          if(itype.eq.2)write(*,*)'Select option "GRAV"ity'
          go to 400
        endif
c
c <EBOD> Input Body
c ---------------
2000    do 2005 ibod=1,maxbod
          if(npts(ibod).eq.0) go to 2010
2005    continue
        write(*,*)'ERROR,the maximum number of bodies(10) already'
        write(*,*)'To enter a new one, one must be deleted.'
        call sound(15,6000)
        go to 400
c
2010    write(*,*)' Body',ibod,' will be created.'
2011    write(*,'(a\)')' Enter no. of points in body(1-19) : '
        read(*,*,err=99000) npt
        if(npt.gt.maxnpt.or.npt.le.2) then
          write(*,*)'No. of points must be from 3 to ',maxnpt
          go to 2011
        endif
        npt = npt + 1
        npts(ibod) = npt
        npt1 = npt - 1
        write(*,*)'Enter ',npt1,' X and Z body point coord. pairs(km);'
        write(*,*)'Note: points must be entered in clockwise order'
        do 2173 i=1, npt1
          write(*,2150) i
2150      format(1h ,'Point ',i3,' X,Y : ',\)
          read(*,*,err=2175) x(i,ibod),z(i,ibod)
2173    continue
        x(npt,ibod) = x(1,ibod)
        z(npt,ibod) = z(1,ibod)
        nbods = nbods + 1
        kalk = 1
        if(iscope.eq.1) call plbod(ibod,-1)
        goto 350
2175    write(*,*)'Input ERROR, body',ibod,' deleted'
        npts(ibod) = 0
        go to 400
```

```
c
c <MPOI> Move point
c ----------
3000  call poin
      goto 350
c
c <EPAR> Enter body parameters
c ----------------------
4000  write(*,'(/a\)')' Enter body number for parameter change : '
      read(*,*,err=99000) ibod
      if(npts(ibod).eq.0) then
        write(*,*)'ERROR, body',ibod,' not defined'
        go to 400
      else
        call param(ibod)
        go to 350
      endif
c
c <CONT> Optimize contrast
c -----------------
5000  call ampl
      goto 350
c
c <DRAW> Draw graphics (full view)
c ----------------------
6000  if(nfield(itype).eq.0) go to 98000
      mode = 1
      xlpl = xpos(1,itype)
      xupl = xpos(nfield(itype),itype)
      zlpl = zmax
      zupl = 0
      if (iscope.eq.1) then
        call planom (0,0)
        if(idiff.lt.0) call plobs
        call plbod (0,127)
      else
        write(*,'(/a/)')' Graphics off, Call "GRAP" to change'
      endif
      goto 400
c
c <SKET> Sketch mode (Draw area specified by zoom limits)
c ----------------------------------------------
7000  if(skxmin.eq.skxmax) then
        write(*,'(/a/)')' Zoom not specified, call option "ZOOM"'
        go to 400
      endif
      if(nfield(itype).eq.0) go to 98000
      mode = 2
      xlpl = skxmin
      xupl = skxmax
      zlpl = skzmax
      zupl = skzmin
      if(iscope.eq.1) then
        call planom (0,0)
        if(idiff.lt.0)call plobs
```

```
        call plbod (0,127)
      else
         write(*,'(/a/)')' Graphics off, call "GRAP" to change'
      endif
      goto 400
c
c <READ> Read model
c ----------
8000  if(name.ne.iblank) then
        itype = 1
        iret = 0
        call readf(iret)
        if(iret.eq.-1) then
          write(*, *) 'Model ', name, ' read'
          if(nfield(itype).eq.0) then
            if(itype.eq.1) then
              write(*,*)'No gravity data, changed to magnetics'
              itype = 2
            else
              write(*,*)'No magnetic data, changed to gravity'
              itype = 1
            endif
          endif
          kalk = 1
          do 8005 ipt=1 , nfield(itype)
8005      xpos(ipt,itype) = xoffs(itype) + (ipt-1)*space(itype)
          call sav
        else
          write(*,*)'WARNING!,model ',name,' NOT read'
            if(iret.eq.0)write(*,*)'Model ',name,' not found'
        endif
      else
        write(*,*)' No model name specified, call option "TNAM"'
        write(*,*)' to list available models.'
      endif
      go to 400
c
c <WRIT> Write model
c -----------
9000  if(name.ne.iblank) then
        call writef(iret)
        if(iret.eq.-1) then
          write(*, *) 'Model ', name, ' written'
        else
          write(*,*) 'WARNING!,model ',name,'NOT written'
        endif
      else
        write(*,*)' WARNING!,model not written,'
        write(*,*) ' No model name specified, call option "NAME"'
      endif
      goto 400
c
c <ANOM> Calculates Anomalies
c ------------------
10000 if(nbods.eq.0) then
```

```
          write(*,*)'No bodies defined,select option "EBOD"'
          go to 400
        endif
        if(nfield(itype).eq.0) go to 98000
        write(*,*)'Enter body no. for anomaly calculation,'
        write(*,'(a\)')' or "0" for total anomaly : '
        read(*,*,err=99000) ibod
        if(ibod.lt.0.or.ibod.gt.nbods) then
          write(*,*)'ERROR,body no. can be from 0-',nbods
          go to 10000
        endif
        if(iscope.eq.1) then
          if(ibod.eq.0) then
            call planom(0,0)
          else
            call planom(ibod,-1)
          endif
          if(idiff.lt.0) call plobs
        else
          if(kalk.eq.1) call calcan
          write(*, *) ' Anomaly calculated'
        endif
        go to 400
c
c <TANO> Prints anomaly
c -----------------
11000 write(*,'(a\)')' Enter body no. to type (0 for all) : '
      read(*,*,err=11000) ibod
        if(ibod.ge.0.and.ibod.le.maxcal) then
          call typano(ibod)
        else
          write(*,*)'ERROR!,invalid body no., retry'
          go to 11000
      endif
      goto 400
c
c <TOBS> Print out observations
c -----------------------
12000 call typobs
      goto 400
c
c <ECOM> Input comments
c --------------
13000 write(*, *) 'Enter comments (up to 80 char) :'
      read(*,13010,err=99000) kommnt
13010 format(8a10)
      goto 400
c
c <TCOM> Output comments
c ---------------
14000 write(*, 14010) kommnt
14010 format(1h, 8a10)
      go to 400
c
c <NAME> Input Model Name
```

```
c ----------------
15000 write(*,'(/a\)')' Enter model name (10 char. max.) : '
      read(*,15010,err=15025) name
15010 format(a10)
      write(*, 15020) name
15020 format(1x, a10)
      go to 400
15025 write(*,*) ' Input ERROR , retry'
      goto 15000
c
c <TNAM> List models in file
c -------------------
16000 iret = 1
      call readf(iret)
      write(*,*)'To read in a model call option "NAME" to'
      write(*,*)'identify the model, followed by "READ".'
      goto 400
c
c <INSE> Insert point
c ------------
17000 call inse
      goto 350
c
c <DPOI> Delete point
c ------------
18000 call dele
      goto 350
c
c <END > Exit program
c ------------
19000 write(*,*)' Did you "write" your final model to disc ?'
      write(*,'(a\)')' "Y" to  END  : '
      call sound(20,200)
      read(*,'(a)',err=99000) ians
      if(ians.eq.'n'.or.ians.eq.'N') go to 400
      write(*,'(/a/)') ' Magrav terminated'
       stop
c
c <TABL> To graphics tablet control
c -----------------
20000 if(nfield(itype).eq.0) go to 98000
      if(iscope.eq.1) then
        call grap
      else
        write(*,*)'Graphics suppressed,call "GRAP" to enable'
      endif
      goto 400
c
c  <RECO> Recover previous step
c ---------------------
21000 call reco
      if(iscope.eq.1) then
        call planom (0,0)
        call plobs
        call plbod (0,127)
```

```
      endif
      goto 400
c
c <DUMP> Dump current data
c ------------------
22000 call tsca
      if(itype.eq.2) then
        write(*,'(/a)')'     MAGNETIC DUMP'
        write(*,'(a)')'     -------------'
        write(*,22010) dip,dec,xton
22010   format(1h0,'Main field dip :',f6.1,/,
     +' Main field declination :',f6.1,/,
     +' Profile +ve X to North angle:',f6.1,/)
      else
        write(*,'(/a)')'      GRAVITY DUMP'
        write(*,'(a)')'      ------------'
      endif
      call typbod(0)
      call typobs
      ibod = 0
      call typano(ibod)
      goto 400
c
c <MAUT> Move point automatically
c -------------
23000 write(*,*)'Select the body containing the point to '
      write(*,'(a\)')' be moved automatically : '
      read(*,*,err=99000) ibod
      if(npts(ibod).eq.0) then
        write(*,*)'ERROR,body',ibod,' not defined'
        call sound(15,6000)
        go to 400
      endif
      call typbod(ibod)
      write(*,'(a\)')' Select point to be moved automatically: '
      read(*,*,err=99000) npt
      if(npt.lt.1.or.npt.gt.npts(ibod)) then
        write(*,*)'ERROR,point not defined'
        call sound(15,6000)
        go to 400
      endif
      write(*,*)'Processing .....'
      call amove(x(npt,ibod),z(npt,ibod),0.0,0.0,2)
      if(iscope.eq.1) then
        call plbod (ibod,-1)
        call planom (0,-1)
      endif
      go to 350
c
c <TPAR> Type out body parameters
c ------------------------
24000 call typar
      goto 400
c
c <MAGN> Change to magnetics mode and enter parameters
```

```
c ----------------------
25000 write(*,'(/a/)')' Modeling mode set to MAGNETICS'
      if(itype.gt.0) then
        write(*,'(a\)')' Change the magnetic mode parameters(y/n) : '
        read(*,'(a)') ians
        if(ians.eq.'n'.or.ians.eq.'N') go to 25050
      endif
      write(*,*)'Enter the angle from geographic North to the '
      write(*,'(a\)')'  positive X (or profile) direction (cw) : '
      read(*,*,err=99000) xton
      xton = amod(xton,360.)
      if(xton.lt.0.) xton = xton + 360.
      write(*,'(a\)')' Enter magnetic field declination : '
      read(*,*,err=99000) dec
      dec = amod(dec,360.)
      if(dec.lt.0.) dec = dec + 360.
      write(*,'(a\)')' Enter magnetic field dip : '
      read(*,*,err=99000) dip
      dip = amod(dip,360.)
      if(dip.lt.0.) dip = dip + 360.
      write(*,'(a\)')' Enter depth offset added to body points(km) : '
      read(*,*,err=99000) zcon(2)
      if(zcon(2).le.0.) zcon(2) = .001
25050 itype = 2
      call rean
      goto 350
c
c <GRAV> Change to gravity mode and enter "zcon"
c -------------------------
26000 write(*,*)'Modeling mode set to GRAVITY'
      if(itype.gt.0) then
        write(*,'(a\)')' Change gravity mode parameter(y/n) : '
        read(*,'(a)') ians
        if(ians.eq.'n'.or.ians.eq.'N') go to 26050
      endif
      write(*,'(a\)')' Enter depth offset added to bodies(km) /; '
      read(*,*,err=99000) zcon(1)
      if(zcon(1).le.0.) zcon(1) = .001
26050 itype = 1
      call rean
      goto 350
c
c <MENU> Menu of Commands
c -----------------
27000 call menu
      goto 400
c
c <ZOOM> Set limits for "sketch"
c ----
28000 if(iscope.ne.1) then
        write(*,*)'Graphics suppressed, call option "GRAP"'
        write(*,*)' to turn graphics on '
        go to 400
      endif
      write(*,'(/a\)')' Enter minimum X for zoom (km) : '
```

```
      read(*,*,err=99000) skxmin
      write(*,'(a\)')' Enter maximum X for zoom (km) : '
      read(*,*,err=99000) skxmax
      write(*,'(a\)')' Enter minimum Z for zoom (km) : '
28011 read(*,*,err=99000) skzmin
      if(skzmin.lt.0) then
        write(*,*)' ERROR!, minimum Z must be greater than 0'
        call sound(15,6000)
        go to 28011
      endif
      write(*,'(a\)')' Enter maximum Z for zoom (km) : '
      read(*,*,err=99000) skzmax
      if(skxmin.ge.skxmax) then
        write(*,*)' ERROR!, invalid X zoom coordinates,retry'
        call sound(15,6000)
        go to 28000
      endif
      if(skzmin.ge.skzmax) then
        write(*,*)' ERROR!,invalid Z zoom coordinates,retry'
        call sound(15,6000)
        go to 28000
      endif
      mode = 1
      write(*,*)' Call option "SKET" to plot selected area.'
      goto 350
c
c <MSCA> Manually set scaling
c ----------
29000 call mscale
      goto 400


c <GRAP> Turn graphics on/off
c ----------------------
30000 call grinit (iscope)
      go to 400
c
c <TBOD> Prints out body points
c ------------
31000 write(*,'(/a\)')' Enter body number to print("0" = all): '
      read(*,*,err=99000) ibod
        if(ibod.ge.0.and.ibod.le.maxbod) then
          call typbod(ibod)
        go to 400
        endif
      write(*,*) 'ERROR!, body number must be from 0-10'
      goto 31000
c
c <TSCA> Prints out scaling parameters
c ----------
32000 call tsca
      goto 400
c
c <OFFS> Recalculate offset
c ------------------
```

```
33000 if(iscope.eq.1) then
        if(nfield(itype).gt.0) then
          write(*,*)'Old offset = ',offset(itype)
          sum = 0.
          do 33050 i = 1, nfield(itype)
            sum = sum + calc(maxcal,i) - obs(itype,i)
33050     continue
          offset(itype) = sum/nfield(itype)
          write(*,*)'New offset = ',offset(itype)
          if(iscope.eq.1.and.idiff.eq.0)call plobs
        else
          write(*,*)'Cannot calculate offset, no observed data'
        endif
      else
        write(*,*)'"OFFS" is applicable only in graphics mode'
      endif
      goto 400
c
c <DIFF> Difference plotting on/off
c ------------------
34000 idiff = -idiff
      if(idiff.gt.0) then
        write(*,*)'Difference mode now ON'
        if(iscope.eq.1) then
          call planom(0,0)
        endif
      else
        write(*,*)'Difference mode now OFF'
        if(iscope.eq.1) then
          call planom(0,0)
          call plobs
        endif
      endif
      goto 400
c
c <HELP> Help text
c ---------
35000 write(*,35010)
35010 format(1h ,
     +'  Magrav is a 2.5 dimensional magnetics and gravity modeling',
     +/,' program. Three modes of operation are possible :',
     +/,'   1) Text mode with no graphics',
     +/,'   2) Graphics enabled with keyboard control',
     +/,'   3) Graphics enabled with graphics tablet control',
     +/,' To start modelling the following options are called',
     +/,' in order : MAGN or GRAV,EOBS,EBOD,EPAR,ANOM.',/)
35020 call menu
      write(*,*)' Additional help is available for each option'
      write(*,'(a\)')' Enter HELP option(<CR> to return) : '
      call what(iwhat)
      if(iwhat.eq.-1) go to 400
      if(iwhat.eq.0) then
        write(*,*)' This option not recognized, try again'
        go to 35020
      endif
```

```
      call help(iwhat)
      go to 35020
      go to 400
c
c <MBOD> Move body
c ----------------
36000 write(*,'(a\)')' Enter number of body to be moved : '
      read(*,*,err=99000) ibod
      if(npts(ibod).eq.0) then
        write(*,*)'ERROR,body',ibod,' not defined'
        call sound(15,6000)
        go to 400
      endif
      call typbod(ibod)
      write(*,'(a\)')' Enter X shift for body(km) : '
      read(*,*,err=99000) xshift
      write(*,'(a\)')' Enter Z shift for body(km) : '
      read(*,*,err=99000) zshift
      do 36010 j = 1 , npts(ibod)
        x(j,ibod) = x(j,ibod) + xshift
        z(j,ibod) = z(j,ibod) + zshift
36010 continue
      if(iscope.eq.1) call plbod (0,127)
      ianom(ibod) = 0
      kalk = 1
      go to 350
c
c <DBOD> Delete body
c ------------------
37000 write(*,'(a\)')' Enter number of body to be deleted : '
      read(*,*,err=99000) ibod
      if(ibod.lt.1.or.ibod.gt.maxbod) then
        write(*,*)'ERROR!,body no. must be from 1 to',maxbod
        call sound(15,6000)
        go to 400
      endif
      if(npts(ibod).eq.0) then
        write(*,*)'ERROR,body',ibod,' not defined'
        call sound(15,6000)
        go to 400
      endif
      call typbod(ibod)
      npts(ibod) = 0
      nbods = nbods - 1
      kalk = 1
      write(*,*) 'Body',ibod,' deleted'
      if(iscope.eq.1) then
        call plbod(0,127)
        call planom(0,-1)
      endif
      go to 350
c
c <ASCL> Automatic scaling for anomaly plot
c ----------------------------------------
38000 if(iscope.eq.1) then
```

```
          call ascale
          call planom(0,0)
          if(idiff.lt.0)call plobs
        else
          write(*,*)'"ASCA" is applicable only in graphics mode'
        endif
        go to 400
c
c Incorrect order branch
c ----------------------
98000 write(*,*)'ERROR, Before this option can be called '
      write(*,*)'  either an model must be "READ" in or'
      write(*,*)'  observed data be read in or entered'
      write(*,*)'  manually using option "EOBS"'
      call sound(15,6000)
      go to 400
c
c input error branch
c ------------------
99000 write(*,'(/a/)')' Input ERROR , retry ;'
      go to 400
      end
```

```
c
c MAGRAV2 SUBROUTINE BLOCK : MS1
c Edited last : Apr.  27 /1986 ; J. Broome
c
$nofloatcalls
c*****************************************************
c  purpose -to calculate the sine of degree in degrees

      function degsin(degree)

      data pi/3.1415926535/

      radian = (pi/180.) * degree
      degsin = sin(radian)
      return
      end
c******************************************
c
c  purpose to calculate the cosine of degree

      function degcos(degree)

      data pi/3.1415926535/

      radian = (pi/180.) * degree
      degcos = cos(radian)
      return
      end

c**************************************************************
c
c  reference: thomas enmark(1981) a versatile interactive
c  computer program for computation and automatic
c  optimization of gravity models; geoexploration,19,47-66.


      function deltag(x1,x,z1,z,y)

      xp = x1 - x
      zp = z1 - z
      a0 = sqrt(xp*xp + zp*zp)
      a1 = 1./a0
      zn = xp*a1
      fi = atan2(zp,xp)
      cof = cos(fi)
      sif = sin(fi)
      u = cof*x + sif*z
      u1 = cof*x1 + sif*z1
      w = -sif*x + cof*z
      r = sqrt(u*u + w*w)
      r1 = sqrt(u1*u1 + w*w)
      ak = (x*z1 - z*x1)/(a0*a0)
      rr = sqrt(r*r + y*y)
      rr1 = sqrt(r1*r1 + y*y)
      ratio1 = (y + rr)/(y + rr1)
```

```
      rlogl = alog(rl*ratiol/r)
      ratio2 = y/rrl
      a = ratio2 * ul/w
      ratio3 = y/rr
      b = ratio3 * u/w
      atl = atan(a)
      at2 = atan(b)
      rlog2 = alog((ul + rrl)/(u + rr))
      deltag = ak * (zp*rlogl + xp*(atl-at2)) + zn*rlog2*y
      return
      end


c***************************************************************************
c
c   subroutine dfs001
c
c     This program minimizes the sum of the squares of non linear functic
c     The method used has been developed and described by M.J.D.Powell
c     in 'The Computer Journal' Vol.7,No.4,Jan.1965, Page 303.
c     The method finds X(1)..........X(N) such that SM is a minimum
c        where SM(X(1)..........X(N)) = SUM
c     over K of (F(K,X(1)............X(N))**2)
c        where K runs from 1 to M with M  greater than N
c
c
c     The parameter names are as follows
c
c     M =  number of observations of the function F
c     N =  number of independant variables
c
c     F() is array of size greater than M, on leaving DFS001 this will
c     contain the values of F(I,X(1)......X(N))  I = 1 to M
c
c     X() is an array of size greater than n which contains the values o
c     the variables X(1) to X(N). On entering DFS001 they are the initial
c     approximations to the minimum. On leaving DFS001 these will be the
c     best values obtained.
c
c     E() is an array of size equal to 2*N. The first N values are the
c      fractional accuracies of the parameters required.
c     I.E. E(I) = 0.0100 requests an accuracy of 1 in X(I).
c     The rest of the array E(N) to E(2*N) is used as working space.
c     Note: E(I) is effectively used as a mesh on the first iteration
c     to form the first derivative of the function F and must therefore
c     be reasonably small escale is a number whose value limits
c     the movement of the variable in any one iteration
c     to an amount equal to ESCALE*E(I)
c
c     IWRITE is an integer which controls the amount of information
c     printed by the routine. there will be a writeout every IWRITE
c     iterations. The writeout consists of the iteration number, number
c     of function evaluations, the values of the variables, the value
c     of the sum of the squares, and the individual function values.
c     If IWRITE is negative, there will be a writeout after every IWRITE
```

```
c     iterations but without the function values.
c     If IWRITE is zero, there will be no printed output.
c
c     MAXITC is an integer which will return control to
c     the calling rout after maxitc iterations.
c
c     On leaving DFS001, the first N*N elements of array W() will contai
c     the variance-covariance matrix elements V(I,J) stored in order
c     V(1,1),V(1,2).....V(1,N),V(2,1)...........V(N,N).
c
c     W() is a working array whose size must be equal to or greater than
c     N + ((M + (3*N/2))*N + 1))
c
c     The program calls two other routines DFS002 and CALFUN
c     which must calculate the function values F for the passed values
c     of the parameters X(1)......X(N).
c
c     Subroutine DFS002 is supplied intact and finds the minimum of a
c     function in one dimension

      subroutine dfs001 (m,n,f,x,e,escale,iwrite,maxitc,w,ierr)

      dimension f(1), x(1), e(1), w(1)

c     in this section we initialize some integer constants for the
c     location of information within the array w

      do 100 i = 1, n
        if(x(i).eq.0.) then
          write(*, *) '* Cannot use zero estimate for parameter ', i
          ierr = 1
          return
        endif
100   continue
      mplusn = m + n
      kst = n + mplusn
      nplus = n + 1
      kinv = nplus * (mplusn + 1)
      kstore = kinv - mplusn - 1
      nn = n + n
      maxfun = (2*n) + 2 + (6*maxitc)
      invar = 1

c     the integer invar is normally set to 1 ,after last iteration it
c     is set to 2 and the variance.co-variance matrix is calculated

c     stores the fractional accuracy requested and then calculates the
c     absolute values of the errors.

      do 110 llm = 1, n
        lln = llm + n
        e(lln) = e(llm)
        e(llm) = e(lln) * x(llm)
110   continue
120   k = nn
```

```
c     this region calculates the first derivatives of the function in
c     the co-ordinate directions and normalizes them such that sum over
c     k of (derivative(i)**2) is unity for all values of i

      iamp = 1
      call calfun (m,n,iamp)

c     stores the initial function values

      do 130 i = 1, m
        k = k + 1
        w(k) = f(i)
130   continue
      iinv = 2
      k = kst
      i = 1
140   x(i) = x(i) + e(i)
      call calfun (m,n,iamp)

c     calculates the function values for  f(x + h) where h = absolute
c     accuracy requested.

      x(i) = x(i) - e(i)
      do 150 j = 1, n
        k = k + 1
        w(k) = 0.0
        w(j) = 0.0
150   continue
      sum = 0.0
      kk = nn

c     calculates values of the derivatives in the coordinate directions
c     sums the individual derivatives

      do 160 j = 1, m
        kk = kk + 1
        f(j) = f(j) - w(kk)
        sum = sum + f(j) * f(j)
160   continue

c     this error condition usually occurs because of a coding error in
c     the subroutine calfun.

      if(sum.le.0) then
        if(iwrite.ne.0) then
          write (*,180) i
180       format(' DFS001: E(',i3,') unreasonably small')
        endif
        ierr = i
        do 190 j = 1, m
          nn = nn + 1
          f(j) = w(nn)
190     continue
        return
```

```
      endif

c     in statement 210 we cancel scaling for calculation of co-variance
c     matrix ,normally we go directly to statement 220

200   if(invar.ne.1) then
        sum = e(i) * e(i)
      endif
220   sum = 1.0  / sqrt(sum)
      j = k - n + i

c     calculates the components of the direction vectors d(i)

      w(j) = e(i) * sum
      do 230 j = 1, m
        k = k + 1
        w(k) = f(j) * sum
        kk = nn + j

c     calculates elements of the normal matrix

        do 240 ii = 1, i
          kk = kk + mplusn
          w(ii) = w(ii) + w(kk) * w(k)
240     continue
230   continue

      iless = i - 1
      igamax = n + i - 1
      incinv = n - iless
      incinp = incinv + 1

c     inverts the one by one matrix  w(1)

      if(iless.gt.0) go to 310
        w(kinv) = 1.0/w(1)
260     if(iinv.eq.1) goto 970
        i = i + 1
        if(i-n.le.0) goto 140
        iinv = 1
c
c     this region is passed through only on iteration 0 and sets up the
c
c     writeing control parameters
c
        ff = 0.0
        kl = nn


c
c     evaluates the sum of the squares of the functions
c
        do 270 i = 1, m
          kl = kl + 1
          f(i) = w(kl)
          ff = ff + f(i) * f(i)
```

```
270     continue
        if(invar.eq.2) goto 1140
        icont = 1
        iss = 1
        mc = n + 1
        ipp = iabs(iwrite) * (iabs(iwrite)-1)
        itc = 0
        ips = 1
        ipc = 0
        goto 970

c     this next region performs the matrix inversion of the normal
c     equation matrix by partitioning. the fully inverted matrix is
c     obtained by repeated passes through this region.
c     an outline of the inversion of a matrix by partitioning is given
c     in  elementary matrix algebra  by  f.e.hohn  p.109
c     in  this routine  b = 1/c  and bb = -b/c

310   if(invar.eq.1) then
        b = 1.0
      else
        b = w(i)
      endif
      do 320 j = nplus, igamax
        w(j) = 0.0
320   continue
      kk = kinv
      do 330 ii = 1, iless
        iip = ii + n
        w(iip) = w(iip) + w(kk) * w(ii)
        jl = ii + 1
        if(jl-iless.le.0) then
          do 340 jj = jl, iless
            kk = kk + 1
            jjp = jj + n
            w(iip) = w(iip) + w(kk) * w(jj)
            w(jjp) = w(jjp) + w(kk) * w(ii)
340       continue
        endif
        b = b - w(ii) * w(iip)
        kk = kk + incinp
330   continue
      b = 1.0/b
      kk = kinv

c
c     calculates and stores the elements of the inverted matrix
c
      do 350 ii = nplus, igamax
        bb = -b * w(ii)
        do 360 jj = ii, igamax
          w(kk) = w(kk) - bb * w(jj)
          kk = kk + 1
360     continue
        w(kk) = bb
```

```
        kk = kk + incinv
350   continue
      w(kk) = b
      goto 260

c     now start an iteration , exact details of the method are found in
c     the paper by powell

420   itc = itc + 1
      if(maxitc-itc.lt.0) then
        if(iwrite.eq.0) goto 1060
        write (*, *) 'The maximum number of allowed iterations ',
     +                       'have been performed'
        goto 1060
      endif
      k = n
      kk = kst

c     initially we calculate the vector p
c       where p(i) = sum over k (gamma(k,i)*f(k))

      do 430 i = 1, n
        k = k + 1
        w(k) = 0.0
        kk = kk + n
        w(i) = 0.0
        do 440 j = 1, m
          kk = kk + 1
          w(i) = w(i) + w(kk)*f(j)
440     continue
430   continue

c     now calculate the movement component q(i) from the inverse deriv-
c     ative matrix and the vector p in the direction d(i)

      dm = 0.0
      k = kinv
      do 450 ii = 1, n
        iip = ii + n
        w(iip) = w(iip) + w(k) * w(ii)
        jl = ii + 1

c     now select the direction to be replaced by delta after this iter-
c     ation. the direction being replaced is stored in kl
c     the value of abs(p(kl).q(kl))is stored in dm

        if(jl-n.le.0) then
          do 460 jj = jl, n
            jjp = jj + n
            k = k + 1
            w(iip) = w(iip) + w(k) * w(jj)
            w(jjp) = w(jjp) + w(k) * w(ii)
460       continue
          k = k + 1
        endif
```

```
          if(dm-abs(w(ii)*w(iip)).lt.0.) then
            dm =  abs(w(ii) * w(iip))
            kl = ii
          endif
450     continue

c       calculates the absolute error requested

        do 465 llm = 1 , n
          lln = llm + n
          e(llm) = e(lln) * x(llm)
465     continue

c       calculate the direction and distance to the minimum, a component
c
c       is delta(i)  =  q(i)*d(i) where d(i) is an n component vector


        ii = n + mplusn * kl
        change = 0.0
        do 470 i = 1, n
          jl = n + i
          w(i) = 0.0
          do 480 j = nplus, nn
            jl = jl + mplusn
            w(i) = w(i) + w(j) * w(jl)
480       continue
          ii = ii + 1
          w(ii) = w(jl)
          w(jl) = x(i)

c       select change to be max(delta(i)/e(i)) i.e. co-ordinate direction
c       whose distance from the minimum is farthest with respect to the
c       requested accuracy.

          if(abs(e(i)*change)-abs(w(i)).le.0.) then
            change =  abs(w(i)/e(i))
          endif
470     continue

        do 490 i = 1 , m
          ii = ii + 1
          jl = jl + 1
          w(ii) = w(jl)
          w(jl) = f(i)
490     continue

        fc = ff
        acc = 0.10/change
        it = 3
        xc = 0.0
        xl = 0.0
        is = 3
        itmax = 6
        relac = 0.1
```

```
      xstep = -amin1(0.5000,escale/change)

c     Selects a grid value of either 0.500 or escale/change to be a
c     minimum. i.e. in DFS002 we search in one dimension along delta
c     for the minimum value of sm in steps of xstep. therefor if escale
c     is small we will move only small distances in direction of delta-
c     if escale is greater than 0.05 * change, then we move just half
c     way to the computed minimum in one step.
c     The condition of change being less than 1.0d0 is accepted as the
c     minimum. in this next part we enter and return from DFS002 in
c     order to determine the value of lambda such that (su(x(i) + lamda*
c     delta(i)) is a minimum DFS002 assumes that sm approaches the min-
c     imum in the form of a quadratic and calculates on this assumption
c     to perform its task DFS001 asks for a minimum of three evalua-
c     tions of the function. if it has not found the minimum in this
c     number, then the value of xstep is changed. a maximum of just six
c     function evaluations is allowed before going on with the calcula-
c     tion. the function evaluations are stored and used to evaluate
c     the function derivatives in the direction delta.

      if(change-1.0.le.0.0) icont = 2
590   call dfs002 (it,xc,fc,itmax,acc,relac,xstep)

c     fc contains intermediate values of the sum of squares
c
c     xc contains the value of lambda
c
c     xl contains intermediate values of lambda

      goto (600,780,780,780), it

600   mc = mc + 1
      if(mc-maxfun.gt.0) then
        if(iwrite.ne.0) then
          write(*, *) 'DFS001 :', maxfun, ' calls of calfun'
        endif
        iss = 2
        goto 780
      endif

c     calculates the function values for x  + lamda(a)*delta and
c     x + lamda(b)*delta and also evaluates sm for these parameter values

630   xl = xc - xl
      do 640 j = 1, n
        x(j) = x(j) + xl * w(j)
640   continue
      xl = xc
      iamp = 0
      call calfun(m,n,iamp)
      fc = 0.0

c     calculates the new sum of squares of the function.

      do 650 j = 1, m
```

```
        fc = fc + f(j) * f(j)
650   continue

      if(is.le.2) goto 690
      k = n
      if(fc-ff) 670, 590, 680

c     sets  fmin to the lowest and fsec to the second lowest sum of
c     squares of the function.

670   is = 2
      fmin = fc
      fsec = ff
      goto 750

680   is = 1
      fmin = ff
      fsec = fc
      goto 750

690   if(fc-fsec.ge.0.0) goto 590
      if(is.ne.2) then
        k = n
      else
        k = kstore
      endif

720   if(fc-fmin) 740, 590, 730
730   fsec = fc
      goto 750
740   is = 3 - is
      fsec = fmin
      fmin = fc

c     stores intermediate values of the parameters.

750   do 760 j = 1, n
        k = k + 1
        w(k) = x(j)
760   continue

c     stores the intermediate function values for the lowest and second
c     lowest sums of squares.

      do 770 j = 1, m
        k = k + 1
        w(k) = f(j)
770   continue

c     at this point we return to DFS002

      goto 590
780   k = kstore

c*********************************************************
```

```
c        this is an alternate exit for this application

       if(itc.eq.maxitc) return

c      we arrive at this point when DFS002 has finished

       kk = n
       if(is.eq.0) then
         k = n
         kk = kstore
       endif
800    sum = 0.0
       dm = 0.0
       jj = kstore

c      we store the new values of x(i), f(i) and the approximate new
c      derivatives.

       do 810 j = 1, n
         k = k + 1
         kk = kk + 1
         jj = jj + 1
         x(j) = w(k)
         w(jj) = w(k) - w(kk)
810    continue

c      again calculate a scaling factor,this time for the new derivative
c      calculates dm as sum over k of  u(k,delta)*f(k,new(x(i)))

       do 820 j = 1, m
         k = k + 1
         kk = kk + 1
         jj = jj + 1
         f(j) = w(k)
         w(jj) = w(k) - w(kk)
         sum = sum + w(jj) * w(jj)
         dm = dm + f(j) * w(jj)
820    continue

       if(iss.eq.2) goto 1060
       j = kinv
       kk = nplus - kl

c      repositions  elements in the normal equation matrix and in the
c      inverted matrix.

       do 830 i = 1, kl
         k = j + kl - i
         j = k + kk
         w(i) = w(k)
         w(k) = w(j-1)
830    continue

       if(kl-n.lt.0) then
         kl = kl + 1
```

```
        jj = k
        do 840 i = kl, n
          k = k + 1
          j = j + nplus - i
          w(i) = w(k)
          w(k) = w(j-1)
840     continue
        w(jj) = w(k)
        b = 1.0/w(kl-1)
        w(kl-1) = w(n)
      else
        b = 1.0/w(n)
      endif
880   k = kinv
      do 890 i = 1, iless
        bb = b * w(i)
        do 900 j = i, iless
          w(k) = w(k) - bb * w(j)
          k = k + 1
900     continue
        k = k + 1
890   continue
      if(fmin-ff.ge.0.) then
        change = 0.0
      else
        ff = fmin
        change =  abs(xc) * change
      endif
930   xl = -dm/fmin
      dum = sum + dm * xl
      if(dum.le.0.) then
        write(*,*)'ERROR in DFS001, dum.le.0'
        ierr = 1
        return
      endif
      sum = 1.0/sqrt(dum)
      k = kstore


c     calculates the components of the new vector direction.

      do 940 i = 1, n
        k = k + 1
        w(k) = sum * w(k)
        w(i) = 0.0
940   continue

c     calculates the new and corrected derivatives of the function in
c     the direction of delta

      do 950 i = 1, m
        k = k + 1
        w(k) = sum * (w(k) + xl*f(i))
        kk = nn + i
c     replaces elements in the matrix i.e. changes the kl direction for
```

```
        do 960 j = 1, n
          kk = kk + mplusn
          w(j) = w(j) + w(kk) * w(k)
960     continue
950   continue

c     go back for new matrix inversion

      goto 310
970   ipc = ipc - iabs(iwrite)

c     the following instruction controls the output of information

      if(ipc.ge.0.0) goto 1040
980   if(iwrite.eq.0) goto 1030
      write(*, 990) itc, mc, ff
990   format(//,5x,'iteration',i4,i9,' calls of calfun',5x,'f=',1pe20.
     +8)
      write(*,1000) (x(i),i = 1,n)
1000  format(5x,'parameters',/,(1p5e20.8))
      if(iwrite.ge.0) then
1010    write (*,1020) (f(i),i = 1,m)
1020    format (5x,'Functions',/,(1p5e20.8))
      endif
1030  ipc = ipp
      if(ips.eq.2) goto 1120
1040  if(icont.eq.1) goto 420
      if(change-1..gt.0.0) goto 1130
1060  if(iwrite) 1070, 1120, 1090
1070  write(*, *) 'DFS001 final values of variables'
      goto 1110
1090  write (*, *) 'DFS001 final values of functions and variables'
1110  ips = 2
      goto 980
1120  invar = 2
      goto 120
1130  icont = 1
      goto 420

c     the statements below store the variance co-variance matrix in the
c     array w

1140  jjvar = kinv - 1
      ff = ff/(m-n)
      do 1150 jvar = 1, n
        do 1160 ivar = jvar, n
          jjvar = jjvar + 1
          jkvar = (jvar-1) * n + ivar
          w(jkvar) = w(jjvar) * ff
1160    continue
        if(jvar-1.gt.0) then
          lvar = jvar - 1
          mvar = 0
          lkvar = lvar * n
          do 1170 kvar = 1, lvar
```

```
            jkvar = lkvar + kvar
            jnvar = (kvar-1) * n + jvar - mvar + kinv - 1
            w(jkvar) = w(jnvar) * ff
            mvar = mvar + kvar
1170      continue
          endif
1150    continue
        jvar = n * n
        return
        end


****************************************************************
c
c   subroutine DFS002
c
c   this subroutine finds the minimum of a function in one
c      dimension. the method used has been described by m.j.d. powell in
c      'the computer journal',vol.7,number 2,july 1964,p.155.
c   the method assumes that the function approaches the minimum
c      quadratically. on first entry it has a value of the function at
c      one point. it then calculates two additional points in the direc-
c      tion requested (returning to the calling routine to do so). the
c      method then predicts the minimum of the  quadratic that passes
c      through the three data points. if the minimum is bracketed by the
c      three data points then minimum predicted is used. if it is not
c      then further steps are taken along the direction requested. a
c      total number of steps maxfun is allowed.
c
c      itest is a control integer which must be set to 2 or 3 on initial
c      entry into the routine and to 1 on subsequent entries during the
c      same search.
c
c      x contains the distance being moved on intermediate returns to
c      the calling routine and the distance to the minimum on the final
c      return f is used to transmit the function values.
c
c      maxfun is the total number of function evaluations allowed , if
c      maxfun is exceeded then routine returns with the nearest value to
c      the minimum in x
c
c      absacc is the absolute accuracy required for the minimum
c
c      relacc is the relative accuracy required for the minimum
c
c      xstep  is the stepping or increment distance in the direction of
c      the minimum

       subroutine dfs002 (itest,x,f,maxfun,absacc,relacc,xstep)


       if(itest.eq.1) goto 7

c      for the first entry into DFS002 itest is set equal to 2 or 3 which
```

```
c      sets up the initial conditions for the computation. itest = 3
c      stores the initial function value and increments x. itest = 2
c      causes a return to the calling routine requesting a first value
c      of the function.
c
c      after setting the intial conditions itest is reset to 1 until the
c      minimum has been obtained.itest  = 2 is a sucessful calculation
c      itest  =  3 is also a sucessful completion ,while itest  =  4 is a
c      return that indicates that maxfun has been exceeded.

       is = 6 - itest
       itest = 1
       iinc = 1
       xinc = xstep + xstep
       mc = is - 3
       if(mc) 9, 9, 6

c      increments function call counter ,if less than maxfun return to
c      calling routine for new function evaluation

2      mc = mc + 1
       if(maxfun-mc.ge.0) return
3      itest = 4
4      x = db
       f = fb
       if(fb-fc.gt.0.0) then
         x = dc
         f = f
       endif
6      return
7      goto (17,15,10,8), is
8      is = 3

c      we come to this point after calculating the first function
c      value for the itest = 2 initial option ,from the next statement (4)
c      on itest  = 3 and itest  = 2 are identical.

c      stores intial function and position values

9      dc = x
       fc = f

c      increments position

       x = x + xstep
       goto 2
10     if(fc-f) 12, 11, 13

c      comes to this point after evaluating second function, goes to 10
c      if first  + second function values are identical, goes to 11 if new
c      one is less and to 9 if new one is larger

11     x = x + xinc

c      in this section nothing is stored as new and old function values
```

```
c     are identical , but x is increment and a new function values is
c     requested , is remains unchanged so we come back to label 7 again

      xinc = xinc + xinc
      goto 2
12    db = x

c     in this section the new function value is larger than the initial
c     one ,again larger one is put into fb , it then changes the sign of
c     xinc and then increments again along x the same amount as for
c     section 11.

      fb = f
      xinc = -xinc
      goto 14
13    db = dc

c     in this section the new function value is less than the initial
c     one. it stores the higher in fb and the lower in fc. x is incre-
c     mented again in the same direction
c     IS is changed to 2 so next entry brings calculation to label 6

      fb = fc
      dc = x
      fc = f
14    x = dc + dc - db
      is = 2
      goto 2
15    da = db

c     arrives in this section having obtained the third function value
c     the largest of the first two is stored in fa and the smallest in
c     fb with the third in fc

      db = dc
      fa = fb
      fb = fc
16    dc = x
      fc = f
      goto 27

c     comes to this section when minimum was within the three function
c     values but not close enough to one end. all the next does is
c     decide which function values to take and in what order so as to
c     minimize rounding errors

17    if(fb-fc.lt.0) go to 21
        if(f-fb.ge.0.0) goto 16
19      fa = fb
        da = db
20      fb = f
        db = x
        goto 27
21    if(fa-fc.gt.0.0) then
        xinc = fa
```

```
          fa = fc
          fc = xinc
          xinc = da
          da = dc
          dc = xinc
        endif
23      xinc = dc
        if((d-db)*(d-dc).lt.0.0) goto 16
24      if(f-fa.lt.0.0) then
          fc = fb
          dc = db
          goto 20
        endif
26      fa = f
        da = x

c       tests to see if the third is smallest. if so, goto 29 and cal-
c       culate second derivative

c       if third function is equal to or larger than the smallest,come to
c       this point and set xinc to twice xstep and sets integer iinc to 2,
c       this integer controls later computations . still calculates the
c       second derivative ,provided that the function values fb and fc are
c       not equal. if they are we goto label 45 where we recalculate with
c       a new x value equal to the mid point of fb and fc which gives a
c       new function value

27      if(fb-fc.le.0.0) then
          iinc = 2
          xinc = dc
          if(fb-fc.eq.0.) goto 40
        endif
29      if((da-db).eq.0) goto 3
        if((da-dc).eq.0) goto 3
        d = (fa-fb)/(da-db) - (fa-fc)/(da-dc)

c       tests sign of second derivative if negative there is a minimum
c       at the calculated value of d , if positive we have found a maximum
c       and goto label 33 to restore calculation for a minimum

c       having calculated a minimum now test for it occuring near enough
c       to the last function evaluation . if it is within either the
c       absolute or relative accuracy then terminate with itest  =  2
c       if not we goto label 36.

        if(d*(db-dc).le.0.0) goto 36
30      d = 0.50 * (db + dc-(fb-fc)/d)
        if(abs(d-x)-abs(absacc).le.0.0.or.
     +    abs(d-x)-abs(d*relacc).le.0.0) then
          itest = 2
          goto 4
        endif
33      is = 1

c       set x to predicted value of minimum ,if this is within the range
```

```
c     examined previously go back with is = 1 and calculate new function
c     value. if outside previous range we recalculate the third function
c     value. if less then 4*increment use predicted value for x. if
c     greater then use 4*increment

      x = d
      if((da-dc)*(dc-d)) 2, 41, 34
34    is = 2
      if(iinc.eq.2) goto 38
      if(abs(xinc)-abs(dc-d)) 37, 2, 2
36    is = 2
      if(iinc.eq.2) goto 39
37    x = dc
      goto 11
38    if(abs(xinc-x)-abs(x-dc).gt.0.0) goto 2
39    x = 0.50 * (xinc + dc)
      if((xinc-x)*(x-dc)) 41, 41, 2
40    x = 0.50 * (db + dc)
      if((db-x)*(x-dc).gt.0.0) goto 2
41    itest = 3
      goto 4
      end
```

```
c
c MAGRAV2 SUBROUTINE BLOCK : MS2
c Edited last : June 19  /1986 ; J. Broome
c
$nofloatcalls
c***********************************************************************
      SUBROUTINE WHAT (IWHAT)
c
c purpose : To interpret text mode commands and return a
c          value in "iwhat" that tells the program what to do
c
c parameters : iwhat - "1-38" for legitimate commands
c                      "0" for unrecognized commands
c                      "-1" for null response
c
$include:'magrav.cmn'
      character ians*4, lckomms(38)*4, uckomms(38)*4
c
      data lckomms/'eobs','ebod','mpoi','epar','cont','draw','sket',
     +    'read','writ','anom','tano','tobs','ecom',
     +    'tcom','name','tnam','inse','dpoi','end','tabl',
     +    'reco','dump','maut','tpar','magn','grav','menu',
     +    'zoom','msca','grap','tbod','tsca','offs','diff','help',
     +    'mbod','dbod','asca'/
c
      data uckomms/'EOBS','EBOD','MPOI','EPAR','CONT','DRAW','SKET',
     +    'READ','WRIT','ANOM','TANO','TOBS','ECOM',
     +    'TCOM','NAME','TNAM','INSE','DPOI',' END','TABL',
     +    'RECO','DUMP','MAUT','TPAR','MAGN','GRAV','MENU',
     +    'ZOOM','MSCA','GRAP','TBOD','TSCA','OFFS','DIFF','HELP',
     +    'MBOD','DBOD','ASCA'/
c
      data nkomms/38/
c
      read(*,150,err=200) ians
150   format(a4)
      if(ians.eq.' ') then
        iwhat = -1
        return
      endif
      do 170 iwhat = 1, nkomms
        if(lckomms(iwhat).eq.ians) return
        if(uckomms(iwhat).eq.ians) return
170   continue
200   iwhat = 0
      return
      end
c
***********************************************************************
      SUBROUTINE POIN
c
c purpose : To allow body point positions to be changed in text mode
c
c
$include:'magrav.cmn'
```

```
c
100   write(*,'(/a)')' Enter the body number for the point'
      write(*,'(a\)')'  to be moved : '
      read(*,*,err=400) ibod
      if(ibod.gt.nbods.or.ibod.le.0)  then
        write(*,*)'ERROR!,body no. ',ibod,' not defined'
        call sound(15,6000)
        goto 100
      endif
      call typbod(ibod)
200   write(*,'(a\)')' Enter no. of point to be changed : '
      read(*,*,err=400) npt
      if(npt.ge.npts(ibod).or.npt.le.0) then
        write(*,*)'ERROR!,point ',npt,' not defined'
        go to 200
      endif
      write(*,1000) npt,x(npt,ibod),z(npt,ibod)
1000  format(1h ,'Point ',i3,',X and Z (km) : ',2f10.2)
      write(*,'(a\)')' Enter new X and Z position (km) : '
      read(*,*,err=400) x(npt,ibod),z(npt,ibod)
      if(z(npt,ibod).lt.0) then
        write(*,*)'Z cannot be less than 0, Z set to 0'
        z(npt,ibod) = 0
      endif
      ianom(ibod) = 0
      kalk = 1
c
      if(npt.eq.1) then
c    If first point changes change last point
        npt = npts(ibod)
        x(npt,ibod) = x(1,ibod)
        z(npt,ibod) = z(1,ibod)
      endif
      if(iscope.eq.1) then
        call plbod(ibod,-1)
      endif
      return
c
400   write(*,*) ' Input ERROR! '
      return
      end
c
**********************************************************************
      SUBROUTINE INSE
c
c purpose : To allow points to be inserted into bodies in text
c             mode
c
$include:'magrav.cmn'
c
100   write(*,'(/a\)')' Enter body no. for inserted point : '
      read(*,*,err=400) ibod
        if(ibod.lt.1.or.ibod.gt.nbods) then
        write(*,*)'ERROR!,body no. ',ibod,' not defined'
        call sound(15,6000)
```

```
          return
        endif
        call typbod(ibod)
200     write(*,*)'Enter the number of the old point after which'
        write(*,'(a\)')' the new point is to be inserted : '
        read(*,*,err=400) npt
        if(npt.lt.npts(ibod)) then
            write(*,'(a\)')' Enter X and Z coordinates(km) : '
            read(*,*,err=400) xxx,zz
            call insert(xxx,zz,ibod,npt,iret)
            if(iret.ne.-1) then
              write(*,*)'ERROR!,inserting point in body'
              call sound(15,6000)
            else
              write(*,*)'Point inserted in body',ibod
            endif
            return
        else
          write(*,*)'ERROR!,old point no. exceeds  number in body'
          call sound(15,6000)
          go to 200
        endif
400     write(*,*)'Input ERROR'
        go to 100
        return
        end
c
*************************************************************************
        SUBROUTINE DELE
c
c purpose : To delete a point from a body in text mode
c
$include:'magrav.cmn'
c
100     write(*,'(/a\)')' Enter body number for point deletion : '
        read(*,*,err=400) ibod
        if(ibod.gt.nbods.or.ibod.lt.1) then
          write(*,*)'ERROR!,body no. ',ibod,' not defined'
          return
        endif
        call typbod(ibod)
200     write(*,'(a\)')' Enter number of point to be deleted : '
        read(*,*,err=400) npt
        if(npt.lt.1.or.npt.ge.npts(ibod)) then
          write(*,*)'ERROR!,point ',npt,' not found'
          call sound(15,6000)
          return
        endif
        xxx = x(npt,ibod)
        zz = z(npt,ibod)
        call delete(ibod,npt,iret)
        if(iret.eq.-1) then
          write(*,*)'point ',npt,' deleted'
        else
          write(*,*)'WARNING!,point ',npt,' not deleted'
```

```
      endif
      return
c
400   write(*,*)'Input ERROR'
      return
      end
c
***********************************************************************
      SUBROUTINE RECO
c
c purpose : To read the common block into the program to recover
c             to the previous step.
c
$include:'magrav.cmn'
c
      if(nback.eq.0) then
        write(*,*)'Recovery impossible,nothing in recovery file'
        call sound(15,6000)
        return
      endif
c
      nrec = 1
      if(nrec.ge.nback) then
        nrec=nback+20-nrec
        nback=nrec
      else
        nback = nback - nrec
      endif
      if(irecov(nback).eq.0) then
        write(*,*)'Recovery impossible,nothing in recovery file'
        call sound(15,6000)
        return
      endif
      read(iscr,rec=nback,err=300) modrec
        write(*, *) 'Model ', name, ' recovered'
      return
300   write(*,*)'WARNING!,model not recovered,iostat= ',kerr
      return

400   write(*,*)'Input ERROR'
      return
      end
c
*******************************************************************
      SUBROUTINE WRITEF(IRET)
c
c purpose : to write model data to lun "model" for  later
c           recovery
c iret : -1 o.k
c        0 eof
c        1 parity
c        2 disallowed duplicate name or parity error on write
c
$include:'magrav.cmn'
c
```

```
c   check for duplicate names
c
      do 100 imod = 1 , nmod
        if(name.eq.ix(imod)) then
          write(*,'(/a\)')' Model already exists,overwrite?,(y/n): '
          read(*,710) ians
710       format(a1)
          if(ians.ne.'y'.and.ians.ne.'Y') then
            iret = 2
            return
          endif
c         Rewrite existing model in place
          go to 1060
        endif
100   continue
c *Add current model to end of models file
c
1000  continue
      nmod=nmod+1
      if(nmod.le.lngix) then
        imod = nmod
        ix(imod) = name
      else
        write(*,*)'Models file full, overwrite existing model'
        write(*,*)' or create a new models file.'
        nmod = lngix
        iret = 2
        return
      endif
1060  do 1050 ibod=1,nbod
1050  ianom(ibod) = 0
      write(model,rec=imod,err=1100,iostat=ierr) moddat
      iret = -1
      return
c
1100   iret=2
       write(*,*)'ERROR writing model,iostat= ',ierr
       return
      end
c
****************************************************************
      SUBROUTINE READF(IRET)
c
c purpose : To read model data from models file and to
c       check for the existance of models by name
c       operation is determined by iret299
c
c  input : iret = 0 search for and find file in name
c                 1 list existing model names
c
c  output : iret = -1 o.k
c                   0 not found
c                   1 parity error

$include:'magrav.cmn'
```

```
      if(iret.eq.0) goto 3000
c
c for input iret=1 list model names
2000  if(nmod.eq.0) then
        write(*,*)'No models defined'
        iret=-1
        return
        endif
      write(*,'(/a/)')' Current model names : '
      write(*,2060)(j,ix(j),j=1,nmod)
2060  format(1h,3x,i4,5x,a10)
      iret = -1
      return
c
c  read appropriate file

3000  if(nback.ne.0) call sav

c     first check to make sure there is such a file
c
      do 3100,imod=1,nmod
3100  if(ix(imod).eq.name) goto 3200
c
      iret = 0
      return
c
3200  read(model,rec=imod,err=3220,iostat=ierr) moddat
      goto 3240
c
3220  iret= 1
      write(*,*)'ERROR reading model,iostat= ',ierr
      return
c
c     check over the models and fix up
c     any open bodies
c      (I hope this is never needed!)
c
3240  do 3300 ibod = 1, nbods
        if(npts(ibod).le.0) goto 3290
        npt = npts(ibod)
c
c   if a body is not closed, close it
c
        if(x(1,ibod).ne.x(npt,ibod).or.z(1,ibod).ne.z(npt,ibod)) then
          if(npt.eq.maxnpt) npt = npt - 1
          npt = npt + 1
          npts(ibod) = npt
          ianom(ibod) = 0
          x(npt,ibod) = x(1,ibod)
          z(npt,ibod) = z(1,ibod)
          write(*, *) 'Body :', ibod, ' closed'
        endif
c
c  if any body has duplicate consecutive points,
```

```
c   delete one of them
c
3250    nptl = npts(ibod) - 1
        do 3270 kount = 1, nptl
          if(x(kount,ibod).eq.x(kount+1,ibod)) then
            if(z(kount,ibod).eq.z(kount+1,ibod)) then
              call delete(ibod,kount,iret)
              if(iret.ne.0) then
                write(*, *) 'Body : ', ibod, ' Point : ', kount,
     +                           ' duplicate and deleted'
                goto 3250
              endif
            endif
          endif
3270    continue
c
        if(npts(ibod).le.2) then
          npts(ibod) = 0
          write(*,*)'Body ',ibod,' deleted,(less than 3 points)'
        endif
c
c reset all body co-ords to (0,0), if body being deleted
c
3290    if(npts(ibod).le.0) then
          do 3295 kount = 1, nptl
            x(kount,ibod) = 0.
            z(kount,ibod) = 0.
3295      continue
        endif
3300  continue
      call rean
      iret = -1
      kalk = 1
      return
      end
c
*********************************************************************
      SUBROUTINE SAV
c
c purpose : To save the current model in case of system crash
c           or user error
c
$include:'magrav.cmn'
c
      if(nback.eq.0) then
        do 10 k = 1 , 20
10      irecov(k) = 0
      endif
      nback= nback+1
      if(nback.gt.20) nback = nback - 20
      write(iscr,rec=nback,err=100,iostat=ierr) modrec
      irecov(nback) = 1
      return
c
100   write(*,*)'ERROR!,writing to scratch file,iostat= ',ierr
```

```
      return
      end
c
c ************************************************************
      SUBROUTINE TYPOBS
c
c Purpose : To print out the observed data
c
$include:'magrav.cmn'
c
      if(itype.eq.1) then
        write(*,'(/a/)')' GRAVITY observations'
      else
        write(*,'(/a/)')' MAGNETIC observations'
      endif
      write(*,1000) offset(itype)
1000  format(' Value added to data for plotting = ',f7.2)
      write(*,'(/a/)')' Point, Position(km), Data value : '
      nf = nfield(itype)
      do 200 i = 1, nf
        write(*,1010) i,xpos(i,itype),obs(itype,i)
1010    format(i4,2x,f10.2,2x,f10.2)
200   continue
      return
      end
c
c *************************************************************
      SUBROUTINE ASCALE
c
c Purpose : to automatically scale anomaly data to fill the
c           screen.
c
$include:'magrav.cmn'
c
      anmin = 999999.9
      anmax = -999999.9
      do 10 ipt = 1 , nfield(itype)
        obsoff = obs(itype,ipt) + offset(itype)
        anmin = amin1(anmin,obsoff)
        anmin = amin1(anmin,calc(maxcal,ipt))
        anmax = amax1(anmax,obsoff)
        anmax = amax1(anmax,calc(maxcal,ipt))
10    continue
      danom = (anmax-anmin)*0.10
      anomax(itype) = anmax + danom
      anomin(itype) = anmin - danom
      write(*,1000) anomin(itype),anomax(itype)
1000  format(1h 'Anomaly plot scale minimum and maximum :',
     +2f10.2)
c
      return
      end
c
c *************************************************************
      SUBROUTINE HELP(IWHAT)
```

```
c
c purpose : To print out descriptive text about each text mode
c          option
c
c parameters : iwhat - input parameter indicating which option
c
      write(*,*)'   '
      if(iwhat.lt.1.or.iwhat.gt.39) return
      goto(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
     +     21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,
     +     38) iwhat
c
1     write(*,*)'EOBS    (Enter OBServed)'
      write(*,*)'    This option allows you to enter the measured'
      write(*,*)'   gravity or magnetic data. Values can be entered'
      write(*,*)'   manually from the keyboard of from a "profil"'
      write(*,*)'   file.'
      return
2     write(*,*)'EBOD     (Enter BODy)'
      write(*,*)'    This option allows you to enter source body'
      write(*,*)'   points. Up to a maximum of 10 bodies with a'
      write(*,*)'   maximum of 19 points are possible. Body numbers'
      write(*,*)'   are selected automatically and EPAR is used to '
      write(*,*)'   enter magnetic or gravity characteristics.'
      write(*,*)'   Point must be entered in clockwise order !'
      return
3     write(*,*)'MPOI     (Move POInt)'
      write(*,*)'    The X and Z coordinates of body points can be'
      write(*,*)'   moved using this option. The point to be moved'
      write(*,*)'   is specified by body number and point number.'
      return
4     write(*,*)'EPAR     (Enter PARameters)'
      write(*,*)'    Body strike extent,density, and magnetization'
      write(*,*)'    can be entered. For density and magnetization,'
      write(*,*)'    a minimum and maximum value specify the range '
      write(*,*)'    for automatic contrast setting with CONT.'
      write(*,*)'    Declination and dip are specified to allow'
      write(*,*)'    remanent magnetization to be accounted for. With'
      write(*,*)'    no remanent magnetization they are set the same'
      write(*,*)'    as the field values.'
      return
5     write(*,*)'CONT    (CONTrast)'
      write(*,*)'    This option automatically varies the density'
      write(*,*)'   or magnetization to improve the least squares'
      write(*,*)'   fit of the calculated anomaly to the measured one.'
      return
6     write(*,*)'DRAW    (DRAW)'
      write(*,*)'    In graphics mode, this option draws the full'
      write(*,*)'   length of both profiles and the bodies on the'
      write(*,*)'   graphics monitor.'
      return
7     write(*,*)'SKET    (SKETch)'
      write(*,*)'    In graphics mode, this option draws a subarea'
      write(*,*)'   of the body display defined using ZOOM.'
      return
```

```
8     write(*,*)'READ    (READ model)'
      write(*,*)'    This option reads a model from the models'
      write(*,*)'   file with the current name.'
      return
9     write(*,*)'WRIT    (WRITe model)'
      write(*,*)'    This option writes the current model to the'
      write(*,*)'   models file with the current name.'
      return
10    write(*,*)'ANOM    (ANOMaly)'
      write(*,*)'    Calculate the anomaly and plot it if you are'
      write(*,*)'   in graphics mode. If body=0 is selected the '
      write(*,*)'   total anomaly is plotted, otherwise a colour-'
      write(*,*)'   coded anomaly for the specified body is plotted.'
      return
11    write(*,*)'TANO    (Type ANOmaly)'
      write(*,*)'    Print out the combined anomaly on the text '
      write(*,*)'   monitor.'
      return
12    write(*,*)'TOBS    (Type OBServed)'
      write(*,*)'    Print out the observed magnetic or gravity data'
      write(*,*)'   on the text monitor.'
      return
13    write(*,*)'ECOM    (Enter COMments)'
      write(*,*)'    This option allows you to enter a 80 character'
      write(*,*)'   text string to describe the model. The string is'
      write(*,*)'   stored with the model on disc.'
      return
14    write(*,*)'TCOM    (Type COMmments)'
      write(*,*)'    This option prints out the text comment stored'
      write(*,*)'   with the model.'
      return
15    write(*,*)'NAME    (NAME)'
      write(*,*)'    This option prompts you to enter a model name'
      write(*,*)'   of up to 10 characters.'
      return
16    write(*,*)'TNAM    (Type NAmes)'
      write(*,*)'    This option prints out the names of models '
      write(*,*)'   in the models file.'
      return
17    write(*,*)'INSE    (INSErt point)'
      write(*,*)'    This option allows you to insert a point in a '
      write(*,*)'   body after a specified point number.'
      return
18    write(*,*)'DPOI    (Delete point)'
      write(*,*)'    This option allws you to delete a specified point'
      write(*,*)'   in a specified body.'
      return
19    write(*,*)'END     (END program)'
      write(*,*)'    Type end to exit from the program. You must call'
      write(*,*)'   WRIT before END to save your new model or any'
      write(*,*)'   changes to your old model or they will be lost.'
      return
20    write(*,*)'TABL    (TABLet)'
      write(*,*)'    This option transfers program control to the'
      write(*,*)'   graphics tablet. Program options can then be '
```

```
      write(*,*)'   selected by placing the cursor over the desired'
      write(*,*)'   option and pressing the cursor button. Body'
      write(*,*)'   points can be identified and moved with cursor'
      write(*,*)'   as well. The small box in the lower left corner'
      write(*,*)'   indicates the cursor positioning tolerance'
      return
21    write(*,*)'RECO    (RECOver)'
      write(*,*)'    This option allows you to go back to previous '
      write(*,*)'   steps in the modelling procedure. Each call of '
      write(*,*)'   RECOver undoes one change. You may recover a'
      write(*,*)'   maximum of 20 changes.'
      return
22    write(*,*)'DUMP    (DUMP model to screen)'
      write(*,*)'    This option prints out all bodies,anomalies,'
      write(*,*)'   parameters and scaling information on the text'
      write(*,*)'   monitor'
      return
23    write(*,*)'MAUT    (Move point AUTomatically)'
      write(*,*)'    This option allows you to automatically move a'
      write(*,*)'   point to improve the least squares fit of the'
      write(*,*)'   calculated to the observed profile. The anomaly'
      write(*,*)'   for the body the point is in will be calculated'
      write(*,*)'   a number of times while moving the point to '
      write(*,*)'   calculate the best position. If the point position'
      write(*,*)'   is not well constrained, The result may not be '
      write(*,*)'   satisfactory due to roundoff errors.'
      return
24    write(*,*)'TPAR    (Type PARameters)'
      write(*,*)'    This option prints out all body density or'
      write(*,*)'   magnetization parameters on the text monitor.'
      return
25    write(*,*)'MAGN    (MAGNetics mode)'
      write(*,*)'    This option puts the program in a magnetic'
      write(*,*)'   modelling mode. The option of setting the magnetic'
      write(*,*)'   field parameters if offered.'
      return
26    write(*,*)'GRAV    (GRAVity mode)'
      write(*,*)'    This option puts the program in a gravity'
      write(*,*)'   modelling mode.'
      return
27    write(*,*)'MENU    (MENU)'
      write(*,*)'    This option prints out the possible options'
      write(*,*)'   on the text monitor.'
      return
28    write(*,*)'ZOOM    (ZOOM)'
      write(*,*)'    This option allows you to set the plotting'
      write(*,*)'   limits for the SKET mode.'
      return
29    write(*,*)'MSCA    (Manual SCAle)'
      write(*,*)'    This option allows you to manually set the '
      write(*,*)'   profile scale parameters,the crossection depth'
      write(*,*)'   and the SKET plotting limits.'
      return
30    write(*,*)'GRAP    (GRAPhics mode)'
      write(*,*)'    This option enables or disables the graphics'
```

```
      write(*,*)'   display. If no graphics monitor is available'
      write(*,*)'   graphics should be disabled. Graphics enabled'
      write(*,*)'   corresponds to MODE 2 of operation. Observed'
      write(*,*)'   profiles are plotted in green and calculated'
      write(*,*)'   profiles in white.'
      return
31    write(*,*)'TBOD    (Type BODies)'
      write(*,*)'    This option types out parameters and point'
      write(*,*)'   positions for a selected body.'
      return
32    write(*,*)'TSCA    (Type SCAle)'
      write(*,*)'    This option prints out scaling parameters'
      write(*,*)'   on the text monitor.'
      return
33    write(*,*)'OFFS    (OFFSet)'
      write(*,*)'    This option automatically calculates the optimum'
      write(*,*)'   least-squares offset for the observed data for'
      write(*,*)'   plotting.'
      return
34    write(*,*)'DIFF    (DIFFerence plot)'
      write(*,*)'    This option switches a automatically scaled plot'
      write(*,*)'   of the difference betweem the observed and '
      write(*,*)'   calculated anomalies on and off.'
      return
35    write(*,*)'HELP    (HELP)'
      write(*,*)'    This option gives a brief description of each'
      write(*,*)'   option. You are in HELP now.'
      return
36    write(*,*)'MBOD    (Move BODy)'
      write(*,*)'    This option allows you to move all points in a'
      write(*,*)'   specified body a specified X and Z distance.'
      return
37    write(*,*)'DBOD    (Delete BODy)'
      write(*,*)'    This option allows you to completely remove a'
      write(*,*)'   specified body.'
      return
38    write(*,*)'ASCA    (Automatic SCAling)'
      write(*,*)'    This option allows automatic scaling of the '
      write(*,*)'   anomalies for plotting.'
      return
      end
c
c ***********************************************************
      SUBROUTINE GRINIT (ISCOPE)
c
c purpose : To initialize the "Halo" graphics functions
c           of the program; read and display a "Halo"
c           format image file on the screen and initialize
c           the drawing colours
c
c parameters : iscope - set = 1, for graphics initialized
c                       set = 2, for  graphics not initialized
c
      character*20 fil,ans
      integer*2 digini,idum
```

```
      logical*2 fex
c
c "halo.dev" is a device driver file provided by halo for
c   your specific graphics board.
c
      if(iscope.eq.1) then
        write(*,*)'Graphics already initialized'
        return
      endif
      write(*,'(/a\)')' Do you want graphics enabled (y/n) : '
      call sound(20,200)
      read(*,'(a)') ans
      if (ans.eq.'n'.or.ans.eq.'N') then
        iscope = 2
        return
      endif
      fil = 'halo.dev'
      inquire(file=fil,exist=fex)
      if(fex.eqv..false.) then
        write(*,*)'ERROR,Graphics device driver file "halo.dev"'
        write(*,*)' must be present on the default drive!'
        write(*,*)' Graphics initialization aborted.'
        call sound (15,6000)
        iscope = 2
        return
      endif
      fil = '"halo.dev"'
      call setdev(fil)
      call inqerr(ifun,ierr)
      if(ierr.ne.0) then
        write(*,*)'ERROR reading "halo.dev" file'
        write(*,*)' Graphics not initialized!'
        iscope = 2
        return
      endif
      call initgr (0)
      call inqerr(ifun,ierr)
      if(ierr.ne.0) then
        write(*,*)'ERROR initializing graphics,'
        iscope = 2
        call sound(15,6000)
        return
      endif
c
c "setcpa" sets the colours used for bodies etc.
c
      call setcpa(0,0,0,0)
      call setcpa(1,255,0,0)
      call setcpa(2,255,255,0)
      call setcpa(3,0,255,255)
      call setcpa(4,255,0,255)
      call setcpa(5,255,127,0)
      call setcpa(6,0,0,255)
      call setcpa(7,255,128,128)
      call setcpa(8,128,128,255)
```

```
      call setcpa(9,128,255,128)
      call setcpa(10,160,160,160)
      call setcpa(11,255,255,255)
      call setcpa(12,0,255,0)
      call setcpa(13,200,128,128)
      call setcpa(127,40,40,40)
      call setcpa(254,120,120,120)
      call setcpa(255,255,255,255)
      call setcol (0)
      call clr
c
c "logo.pic" is a "Halo" image file displayed at the start
c  of the session.
c
      fil = '"logo.pic"'
      call gread (fil)
      call setiee (1)
      idum = digini(idum)
      write(*,'(/a)')' Graphics  system  initialized  for'
      write(*,'(a/)')' Graphics Card matching chosen HALO.DEV file'
      iscope = 1
      return
      end
c
c *************************************************************
      SUBROUTINE MENU
c
c purpose : To print out a list of the available text options
c
      write(*,1000)
1000  format(1h0,'      Current Menu of Options',/,
     +'       -----------------------',/,
     +' ANOM  ASCA  CONT  DBOD  DIFF  DPOI  DRAW  DUMP  EBOD  ECOM',/,
     +'  END  EOBS  EPAR  GRAP  GRAV  HELP  INSE  MAGN  MAUT  MENU',/,
     +' MBOD  MPOI  MSCA  NAME  OFFS  READ  RECO  SKET  TABL  TANO',/,
     +' TBOD  TCOM  TOBS  TNAM  TPAR  TSCA  WRIT  ZOOM',/)
      return
      end
c
c ****************************************************************
c
      SUBROUTINE EOBSE
c
c purpose : To allow the user to enter observed anomaly data
c           from the keyboard.
c
$include:'magrav.cmn'
c
      write(*,'(a\)')' Enter X profile offset (km) : '
      read(*,*,err=900) xoffs(itype)
10    write(*,'(a\)')' Enter profile sampling interval (km) : '
      read(*,*,err=900) space(itype)
      if(space(itype).le.0) then
        write(*,*)'ERROR!,sampling interval must be > 0'
        call sound (15,6000)
```

```
        go to 10
      endif
20    write(*,'(a\)')' Enter No. of values in profile : '
      read(*,*,err=900) nfield(itype)
      if(nfield(itype).gt.maxobs.or.nfield(itype).le.0) then
        write(*,*)'ERROR!,no. of values must be less than',maxobs
        call sound (15,6000)
        go to 20
      endif
      do 30 ipt=1 , nfield(itype)
30    xpos(ipt,itype) = xoffs(itype) + (ipt-1)*space(itype)
      xlen(itype) = space(itype) * (nfield(itype)-1)
      write(*,*)'Profile X length = ',xlen(itype),' km.'
      write(*,*)'Enter reading no. and measured anomaly value'
      write(*,*)' (enter 0,0 to quit)'
100   write(*,'(a\)')' no.,value : '
      read(*,*,err=100) j,ans
      if(j.eq.0) go to 200
      if(j.le.nfield(itype)) then
        obs(itype,j) = ans
        go to 100
      endif
      write(*,*)'ERROR!, ',j,' exceeds maximum of ',nfield(itype)
      go to 100
c
200   if(iscope.eq.1.and.idiff.lt.0) then
        call ascale
        xlpl = xpos(1,itype)
        xupl = xpos(nfield(itype),itype)
        call plobs
        if(nbods.gt.0) call plbod(0,127)
      endif
      return
c
900   write(*,*)'ERROR!, Input error.'
      call sound (15,6000)
      return
      end
c
c ***************************************************************
c
      SUBROUTINE ROBSE(IERR)
c
c purpose : To read observed anomaly data from a profile file.
c
c parameters : ierr - set to "1" for errors reading from file
c
c file format : record 1 - nfield(itype)
c               record 2 - space(itype)
c               record 3 to nfield+2 -obs(itype,ipt)
c                 (All data free formatted)
c
      character*50 pfile
$include:'magrav.cmn'
c
```

```
      write(*,'(/a\)')' Enter profile file filespec : '
      call sound(20,200)
      read(*,'(a)') pfile
      open(10,file=pfile,status='old',form='formatted')
c
      read(10,*,err=999) nfield(itype)
      read(10,*,err=999) space(itype)
      do 100 ipt = 1 , nfield(itype)
        read(10,*,err=999) obs(itype,ipt)
100   continue
      write(*,1000) nfield(itype),space(itype)
1000  format(1h ,'No. of profile values read in : ',i4,/
     +' Profile value sampling interval (km): ',f10.5)
150   write(*,'(a\)')' Enter X profile offset (km) : '
      call sound(20,200)
      read(*,*,err=900) xoffs(itype)
c
      do 200 ipt=1, nfield(itype)
200   xpos(ipt,itype) = xoffs(itype) + (ipt-1)*space(itype)
      xlen(itype) = space(itype)*(nfield(itype)-1)
      write(*,*)'Profile X length = ',xlen(itype),' km.'
      if(iscope.eq.1.and.idiff.lt.0) then
        call ascale
        xlpl = xpos(1,itype)
        xupl = xpos(nfield(itype),itype)
        call plobs
      endif
      ierr = 0
      return
c
900   write(*,*)'Input ERROR!, retry.'
      call sound(15,6000)
      go to 150
c
999   write(*,*)'ERROR!,reading profile file'
      call sound(15,6000)
      ierr = 1
      return
      end
```

```
c
c MAGRAV2 SUBROUTINE BLOCK : MS3
c Edited last : June 20 / 1986 ; J. Broome
c
$nofloatcalls
************************************************************************
      SUBROUTINE REAN
c
c purpose : To zero arrays "calc" and "ianom"

$include:'magrav.cmn'

      do 100 i=1, maxcal
        do 110 j = 1, maxobs
          calc(i,j) = 0.
110     continue
100   continue
      do 200 i = 1, maxbod
        ianom(i) = 0
200   continue
      kalk = 1
      return
      end
c
******************************************************************
      SUBROUTINE TYPBOD (IBOD)
c
c purpose : To type body coordinates and parameters
c
c parameters : ibod = "0" to print all bodies
c                     body number
c
$include:'magrav.cmn'

      character*10 color(10)
      data color/'       red','    yellow',' turquoise',
     +           '    purple','    orange','      blue',
     +           '      pink','light blue','     green',
     +           '      gray'/
c
      if(nbods.eq.0) then
        write(*,*)'No bodies defined!'
        return
      endif
c
      if(ibod.eq.0) then
        ibegin = 1
        iend = maxbod
      else
        ibegin = ibod
        iend = ibod
        if(npts(ibod).eq.0) then
          write(*,*)'Body ',ibod,' not defined.'
          return
        endif
```

```
      endif

      do 100 i = ibegin, iend
        if(npts(i).ne.0) then
          write(*,'(/a,i3)')' BODY :',i
          write(*,*)'colour = ',color(i)
          write(*, *) 'Half-strike length :', bdy(i)
          if(itype.eq.1)
     +      write(*, 270) rmmin(i,1), rhomag(i,1), rmmax(i,1)
270         format(5x,'Min.,Actual, and Max. Density :',/,3f10.2)
          if(itype.eq.2)
     +      write(*, 280) rmmin(i,2), rhomag(i,2),
     +                            rmmax(i,2)
280       format(5x,'Min.,Actual, and Max. Magnetization :',/,
     +    3f10.2)
          write(*,290) bdec(i),bdip(i)
290       format(5x,'Declination=',f5.1,' ,Dip=',f5.1)
          nptl = npts(i) - 1
        write(*,*)'Point no., X,          Z'
          do 110 j = 1, nptl
            write(*,'(i5,2f10.2)') j, x(j,i), z(j,i)
110       continue
        endif
100   continue
      write(*, *) ' '
      return
      end


*****************************************************************
      SUBROUTINE TYPAR
c
c Purpose : to print out magnetization or density parameters
c           for each body.

$include:'magrav.cmn'

c  gravity
      if(itype.ne.2) then
        write(*,*)'Body   minimum    actual   maximum '
        write(*,*)'       density   density   density'
        do 100 i = 1, maxbod
          if(npts(i).gt.0) then
            write(*, 150) i, rmmin(i,1), rhomag(i,1), rmmax(i,1)
150         format(i4,3f10.2)
          endif
100     continue
        return
      endif

c  magnetics
2000  write(*,2100)
2100  format(' Body  minimum    actual   maximum   dec.  dip.',/,
     +'                 magnetization')
```

```
      do 200 i = 1, maxbod
        if(npts(i).gt.0) then
          write(*, 151) i, rmmin(i,2), rhomag(i,2),
     +                    rmmax(i,2), bdec(i), bdip(i)
151       format (i4,3f10.2,2f6.1)
        endif
200   continue
      return
      end

*************************************************************
      SUBROUTINE TYPANO (JBOD)
c
c purpose : To print out anomaly values for body "ibod"

$include:'magrav.cmn'

      ibod = jbod
      if(jbod.eq.0)ibod = maxcal
      if(ibod.lt.maxcal) write(*, *) 'Anomaly Body :', ibod
      if(ibod.eq.maxcal) write(*, *) 'Total Anomaly'
      nf = nfield(itype)
      do 500 i = 1, nf
        if(calc(ibod,i).le.-.00001.or.calc(ibod,i).ge..00001) then
          write(*,400) i,xpos(i,itype),calc(ibod,i)
400       format(i3,1x,f10.2,1x,f10.2)
        endif
500   continue
      return
      end

c*************************************************************
      SUBROUTINE CALCAN
c
c purpose : To calculate the gravity or magnetic anomalies
c           for any bodies that have had their parameters or
c           points changed.(ianom(ibod) set to "0")

$include:'magrav.cmn'

      do 100 i = 1, maxobs
        calc(maxcal,i) = 0.
100   continue

      do 200 ibod = 1, maxbod
        if(npts(ibod).gt.0) then
          if(ianom(ibod).ge.0) then
            if(itype.eq.2) call mag(ibod)
            if(itype.eq.1) call gravc(ibod)
          endif
          nf = nfield(itype)
          do 210 i = 1, nf
            calc(maxcal,i) = calc(maxcal,i) + calc(ibod,i)
210       continue
        endif
```

```
200   continue
      kalk = 0
      return
      end

c*******************************************************************
      SUBROUTINE CHECK (NUMBER,XXX,ZZ,IDUPBD,IDUPPT,IRET)
c
c purpose : This subroutine checks points to see if they are in
c      approximately the same place as existing points. If so
c      the body number and body point number are returned. If
c      not "iret" is set to -1 and the routine is exited.
c
c parameters : number - The "number"th duplicate point
c                       is checked for.
c              xxx    - X coordinate of point to be checked
c              zz     - Z coordinate of point to be checked
c              idupbd - body number for "number"th duplicate
c                       point
c              iduppt - point number body "idupbd"
c              iret   - 0 for duplicated point
c                      -1 for unique point
c

$include:'magrav.cmn'

      numdup = 0
      do  100 idupbd = 1, maxbod
        npt = npts(idupbd)
        if(npt.gt.0) then
          do 90 iduppt = 1, npt
            if(abs(xxx-x(iduppt,idupbd)).gt.xdis) goto 90
            if(abs(zz-z(iduppt,idupbd)).gt.zdis) goto 90

c     duplicate point
            numdup = numdup + 1
            if(numdup.lt.number) goto 90
            iret = 0
            xxx = x(iduppt,idupbd)
            zz = z(iduppt,idupbd)
            return
90        continue
        endif
100   continue

c  unique point
      iret = -1
      idupbd = 0
      iduppt = 0
      return
      end

c*****************************************************************
      SUBROUTINE INSERT (XXX,ZZ,IBOD,NPT,IRET)
c
```

```
c purpose : To insert a new point with coordinates (xxx,zz) after
c           point "npt" in body "ibod".
c
c parameters: xxx    - X coordinate of new point
c             zz     - Z coordinate of new point
c             ibod   - number of body to insert point into
c             npt    - point in body after which point is to
c                      be inserted.
c             iret   - "-1" if point inserted
c                      "0" if point not inserted
c
c
$include:'magrav.cmn'

      iret = 0
      if(ibod.lt.1) return
      if(npt.gt.npts(ibod)) return
      if(npts(ibod).ge.maxnpt) then
        write(*,*)'ERROR,maximum number of points in body'
        call sound(15,6000)
        return
      endif
      iend = npts(ibod)
      npt = npt + 1
      npts(ibod) = npts(ibod) + 1
      if(npt.ne.npts(ibod)) then
        l = npts(ibod)
        do 100 i = npt, iend
          x(l,ibod) = x(l-1,ibod)
          z(l,ibod) = z(l-1,ibod)
          l = l - 1
100     continue
      endif
      ianom(ibod) = 0
      x(npt,ibod) = xxx
      z(npt,ibod) = zz
      iret = -1

c     set last point  =  first just to be sure

      x(iend+1,ibod) = x(1,ibod)
      z(iend+1,ibod) = z(1,ibod)
      kalk = 1
      if(iscope.eq.1) call plbod(ibod,-1)
      return
      end

c****************************************************************
      SUBROUTINE DELETE (IBOD,NPT,IRET)
c
c purpose : To delete point "npt" from body "ibod"
c
c parameters : ibod   - body point is to be deleted from
c              npt    - number of point to be deleted
c              iret   - "0" if point is not deleted
```

```
c                          "-1" if point is deleted
c
$include:'magrav.cmn'

      iret = 0
      if(npts(ibod).eq.0) then
        write(*,*)'ERROR, body',ibod,' not defined'
        call sound(15,6000)
        return
      endif
      if(npt.lt.1.or.npt.gt.npts(ibod)) then
        write(*,*)'ERROR,point not defined'
        call sound(15,6000)
      endif
      iend = npts(ibod) - 1
      npts(ibod) = iend
      if(npt.lt.iend) then
        do 100 i = npt, iend
          ii = i + 1
          x(i,ibod) = x(ii,ibod)
          z(i,ibod) = z(ii,ibod)
100     continue
      endif
      ianom(ibod) = 0
      x(iend,ibod ) = x(1,ibod)
      z(iend,ibod ) = z(1,ibod)
      iret = -1
      kalk = 1
      if(iscope.eq.1) call plbod(ibod,-1)
      return
      end

c*******************************************************************
      SUBROUTINE MAG(IBOD)
c
c purpose : To calculate the magnetic anomaly for body "ibod"
c
c  sources bio computer note 66-1-c april 1966
c   program mag written for pdp-11 by d.heffler,agc,bio 19
c   cdc3150 fortran program mag2new,agc,bio,197...
c
c  modified for 2.5 dimensional bodies by Franca Lindia
c  using equations published by Shuey and Pasquale(1973)
c  in the journal "Geophysics".

$include:'magrav.cmn'

      complex zi,zi1,zi2,yi,fn1,fn2,fn,yr1c,qpx,qpz,qxsm,qzsm,czero
      complex rsum,x21zi
c
      write(*,*)'Calculating mag. an. for body',ibod
      nf = nfield(itype)
      if(rhomag(ibod,2).eq.0.) return
      cdipd = degcos(dip)
      sdipd = degsin(dip)
```

```
      sdd =   degcos(xton-dec)
      cdip =  degcos(bdip(ibod))
      sdip =  degsin(bdip(ibod))
      sd =    degcos(xton-bdec(ibod))
      cdy =   degcos(90.-(xton-bdec(ibod)))
      sdt =   degsin(xton-dec)
      cdipsd = cdip*sd
      cdpcdy = cdip*cdy
      rhobod = rhomag(ibod,2) * 2.0
      y = bdy(ibod)
      ysq = y**2
      yd = 1.0/ysq
      yi = cmplx(0.,yd)
      npt = npts(ibod)
      czero = cmplx(0.,0.)

c   check each field point

      do 3100 k = 1, nf
        qxsm = czero
        qzsm = czero
        rsum = czero
        x1 = x(1,ibod) - xpos(k,itype)
        z1 = z(1,ibod) + zcon(2)
        if(z1.le.0.) goto 9999
        r1 = sqrt( x1**2 + z1**2 + ysq )
        zi1 = cmplx(0.,z1)
        do 3000 j = 2, npt
          x2 = x(j,ibod) - xpos(k,itype)
          z2 = z(j,ibod) + zcon(2)
          if(z2.le.0.) goto 9999

c   if 2 points the same check the point after

          if(x1.eq.x2.and.z1.eq.z2) goto 3000
          z21 = z2 - z1
          x21 = x2 - x1
          zi = cmplx(0.,z21)
          x21zi = x21 + zi
          zi2 = cmplx(0.,z2)
          r2 = sqrt( x2**2 + z2**2 + ysq )
          fn1 = x21zi/(x1+zi1) * (1.0 + r1/y)
     +      +  yi* (x1*z21 - z1*x21)
          fn2 = x21zi / (x2+zi2) * (1.0 + r2/y)
     +      +  yi*(x2*z21 - z2*x21)

c  top and bottom of log >0. since "zcon" not = 0

          if(fn1.eq.czero) goto 9999
          if(fn2.eq.czero) goto 9999
          fn = fn2/fn1
          yr1c = clog(fn)
          qpx = zi/x21zi * yr1c
          qpz =  -x21/x21zi * yr1c
          qxsm = qxsm + qpx
```

```
          qzsm = qzsm + qpz
          rsum = rsum + yrlc
          xl = x2
          zl = z2
          zil = zi2
          rl = r2
3000    continue
        qtot = real(qxsm)
        pxtot = aimag(qxsm)
        pztot = aimag(qzsm)
        rytot = aimag(rsum)
        h =  cdipsd*pxtot + sdip*qtot
        v = cdipsd*qtot - sdip*pztot
        hy = cdpcdy*rytot
        calc(ibod,k) = (v*sdipd + (h*sdd - hy*sdt)*cdipd) * rhobod
3100  continue
      ianom(ibod) = -1
      return

c    error

9999  continue
      write(*, *) 'Body :', ibod, ' Point :', k
      write(*, *) ' Cannot be calculated with present algorithm'
      write(*, *) ' Value out of range '
      write(*, *) ' Anomaly set to zero'
      write(*, *) ' Use "GRAV" or "MAGN" command'
     +           ,' to set a larger Z constant'
      do 10000 k = 1, nf
        calc(ibod,k) = 0.0
10000 continue
      ianom(ibod) = 0
      return
      end

c*******************************************************************
      SUBROUTINE GRAVC(IBOD)
c  purpose-to calculate the gravitational anomaly for body ibod

$include:'magrav.cmn'

      write(*,*)'Calculating anomaly for body ',ibod
      nptl = npts(ibod) - 1
      nf = nfield(itype)
      do 100 k = 1, nf
        calc(ibod,k) = 0.
100   continue
      if(rhomag(ibod,1).eq.0.) return
      y = bdy(ibod)
      do 1000 k = 1, nf
        sum = 0.
        dist = xpos(k,itype)
        xj = x(1,ibod) - dist
        zj = z(1,ibod) + zcon(1)
        if(zj.le.0.) goto 9999
```

```
        do 200 j = 1, nptl
          xjl = x(j+1,ibod) - dist
          zjl = z(j+1,ibod) + zcon(1)
          if(zjl.le.0.) goto 9999
          sum = sum + deltag(xjl,xj,zjl,zj,y)
          xj = xjl
          zj = zjl
200     continue
        calc(ibod,k) = 13.346 * sum * rhomag(ibod,1)
1000  continue
      ianom(ibod) = -1
      return

9999  continue
c      if(iscope.eq.1) call bell
c      if(iscope.eq.1) call anmode
      write(*, *) 'Body : ', ibod, ' Point : ', k, ' is negative'
      do 10000 k = 1, nf
        calc(ibod,k) = 0.
10000 continue
      ianom(ibod) = 0
      return
      end
c
c*************************************************************
      SUBROUTINE PARAM (IBOD)
c
c purpose : To allow body magnetization parameters and strike
c           extent to be entered.
c
c parameters : ibod - body number for parameter change

$include:'magrav.cmn'

      write(*,*)'Current body strike extent (km) = ',bdy(ibod)
10    write(*,'(a\)')' Enter new strike extent (km) : '
      read(*,*,err=900) bdy(ibod)
      if(bdy(ibod).le.0) then
        write(*,*)'ERROR!,strike extent must be greater than 0'
        call sound(15,6000)
        go to 10
      endif
      if(itype.eq.1) then
        write(*,1010) rmmin(ibod,itype),rhomag(ibod,itype),
     +rmmax(ibod,itype)
1010  format(1h ,'Minimum,body, and maximum density contrasts =',/,
     + 3f10.2)
        write(*,'(a\)')' Enter minimum density contrast(g/cc): '
      else
        write(*,1020) rmmin(ibod,itype),rhomag(ibod,itype),
     +rmmax(ibod,itype)
1020  format(1h ,'Minimum,body, and maximum magnetizations =',/,
     +3f10.2)
        write(*,'(a\)')' Enter minimum magnetization (X10-5 emu): '
      endif
```

```
      call sound(20,200)
      read(*,*,err=900) rmmin(ibod,itype)
c
      if(itype.eq.1) then
        write(*,'(a\)')' Enter body density contrast(g/cc): '
      else
        write(*,'(a\)')' Enter body magnetization (X 10-5 emu) : '
      endif
      call sound(20,200)
      read(*,*,err=900) rhomag(ibod,itype)
c
      if(itype.eq.1) then
        write(*,'(a\)')' Emter maximum density contrast(g/cc): '
      else
        write(*,'(a\)')' Enter maximum magnetization (X10-5 emu): '
      endif
      call sound(20,200)
      read(*,*,err=900) rmmax(ibod,itype)
c
      if (itype.eq.2) then
      write(*,1030)dec,dip
1030  format(1h ,'Field declination = ',f8.1,/
     +,' Field dip          = ',f8.1)
        write(*,'(a\)')' Enter body magnetization declination : '
        call sound(20,200)
        read(*,*,err=900) bdec(ibod)
        write(*,'(a\)')' Enter body magnetization dip : '
        call sound(20,200)
        read(*,*,err=900) bdip(ibod)
      endif
c
      kalk = 1
      ianom(ibod) = 0
      return
c
900   write(*,*)'Input ERROR'
      call sound(15,6000)
      return
      end

c*****************************************************
      SUBROUTINE TSCA
c
c purpose : To type out scaling parameters ,etc.

$include:'magrav.cmn'

      write(*, *) 'Model : ', name
      if(itype.eq.1) write(*, *) 'Gravity'
      if(itype.eq.2) write(*, *) 'Magnetics'
      write(*, 46) kommnt
  46  format(' Comments '/,1x,8a10)
      write(*,*)'X length of profil(km) =',xlen(itype)
      write(*,*)'Crossection depth (km) =',zmax
      write(*,*)'X profile offset (km)  =',xpos(1,itype)
```

```
      write(*,*)'Sampling interval(km)  =',space(itype)
      write(*,*)'No. of readings        =',nfield(itype)
      write(*,*)'Anomaly scale minimum  =',anomin(itype)
      write(*,*)'Anomaly scale maximum  =',anomax(itype)
      write(*,*)'Difference scale min.  =',difmin(itype)
      write(*,*)'Difference scale max.  =',difmax(itype)
      write(*,*)'Observed offset        =',offset(itype)
      write(*, *) 'Number of bodies     =',nbods
      if(itype.eq.2) write(*, 210) dec, dip, xton, zcon(itype)
210   format(8x,'dec = ',f8.2,8x,' dip = ',f8.2/
     +' X to n angle : ',f8.2/8x,' Z constant : ',f8.4)
      if(iscope.eq.2) write(*,*)' Graphics suppressed'
      if(mode.eq.2) write(*, *) 'Sketch mode'
      if(mode.eq.1) write(*, *) 'Draw mode'
      write(*, 227) skxmin, skxmax, skzmin, skzmax
227   format(' Sketch limits X:',2f10.2,/13x,   ' Z:',2f10.2)
      write(*, *) 'Maximum number of points per body = ', maxnpt
      write(*, *) 'Maximum number of bodies          = ', maxbod
      write(*, *) 'Maximum profile length            = ', maxobs
      return
      end

******************************************************************
      SUBROUTINE AMPL
c
c purpose : To automatically adjust the density or magnetization
c           contrasts of the bodies to improve the fit of the
c           calculated profile to the measured data.

$include:'magrav.cmn'

      data iwrite/0/, maxit/1/

      ibr = 0
      esc = 1.e4
      if(itype.eq.1) esc = 100.
      nf = nfield(itype)
      if(nf.eq.0) then
        write(*,*)'ERROR, no. observed data '
        call sound(15,6000)
        return
      endif
      if(itype.eq.1) then
        write(*,*)'Old density contrasts for each body : '
      else
        write(*,*)'Old magnetization contrasts for each body :'
      endif
      do 50 j=1 , maxbod
        if(npts(j).ne.0)write(*,'(i5,3x,f10.2)')j,rhomag(j,itype)
50    continue
      call calcan
      nc = 0
      do 100 i = 1, maxbod
        if (npts(i).eq.0)  go to 100
        if(abs(rhomag(i,itype)).ge.1.e-10) then
```

```
          do 110 j = 1, nf
            calc(i,j) = calc(i,j)/rhomag(i,itype)
110       continue
          nc = nc + 1
          xx(nc) = rhomag(i,itype)
          e(nc) = .001
        endif
100   continue
      if (nc.eq.0) then
        write(*,*)'ERROR, no bodies defined'
        call sound(15,6000)
        kalk = 1
        return
      endif
      ierr = 0
      call dfs001(nf,nc,f,xx,e,esc,iwrite,maxit,w,ierr)
      if(ierr.ne.0) then
        write(*,*)'ERROR,in DFS001(err=',ierr,')'
        call sound(15,6000)
        kalk = 1
        return
      endif
      nc = 0
      do 200 i = 1, maxbod
        if(npts(i).eq.0) go to 200
        if(abs(rhomag(i,itype)).ge.1.e-10) then
          nc = nc + 1
          rhomag(i,itype) = xx(nc)
        endif
200   continue
      do 300 i = 1, nf
        calc(maxcal,i) = 0.
300   continue
      do 400 i = 1, maxbod
        if(npts(i).eq.0) go to 400
        do 410 j = 1, nf
          calc(i,j) = calc(i,j)*rhomag(i,itype)
          calc(maxcal,j) = calc(maxcal,j) + calc(i,j)
410     continue
400   continue
      kalk = 0
      if(itype.eq.1) then
        write(*,*)'New density contrasts for each body :'
      else
        write(*,*)'New magnetization contrasts for each body :'
      endif
      do 500 j = 1 , maxbod
        if(npts(j).ne.0)write(*,'(i5,3x,f10.2)')j,rhomag(j,itype)
500   continue
      return
      end

************************************************************************
      SUBROUTINE CALFUN (M,N,IAMP)
C
```

```
c purpose : This program is called by subroutine "DFS001" to
c          calculate the values in array "f" .Array "f" contains the
c          difference between the calculated and observed profiles.
c
c parameters : m    - number of points in the profile
c              n    - number of independent variables ;
c                       (one point is 2 variables ,x and z)
c           iamp    - "1" for point position movement
c                       "0" for contrast optimization
c

$include:'magrav.cmn'

      do 100 i = 1, m
        calc(maxcal,i) = 0.
100   continue
      if(ibr.ne.1) then
        nc = 0
        do 200 i = 1, maxbod
        if(npts(i).eq.0) go to 200
          if(abs(rhomag(i,itype)).ge.1.e-10) then
            nc = nc + 1
            if(iamp.ne.1) then
              if(xx(nc).lt.rmmin(i,itype)) xx(nc) = rmmin(i,itype)
              if(xx(nc).gt.rmmax(i,itype)) xx(nc) = rmmax(i,itype)
            endif
            do 220 j = 1, m
              calc(maxcal,j) = calc(maxcal,j) + calc(i,j) * xx(nc)
220         continue
          endif
200   continue
      else
        do 300 i = 1, 2
          if(ib(i).ne.0) then
            x(np(i),ib(i)) = xx(1)
            z(np(i),ib(i)) = abs(xx(2))
            ianom(ib(i)) = 0
          endif
300     continue
        do 400 i = 3, 4
          if(ib(i).ne.0) then
            x(np(i),ib(i)) = xx(3)
            z(np(i),ib(i)) = abs(xx(4))
            ianom(ib(i)) = 0
          endif
400     continue
        do 500 ibod = 1, maxbod
        if(npts(ibod).eq.0) go to 500
          if(npts(ibod).gt.0) then
            if(ianom(ibod).ge.0) then
              if(itype.eq.2) call mag(ibod)
              if(itype.eq.1) call gravc(ibod)
              ianom(ibod) = -1
            endif
            nf = nfield(itype)
```

```
          do 520 i = 1, nf
            calc(maxcal,i) = calc(maxcal,i) + calc(ibod,i)
 520      continue
        endif
 500  continue
      endif
      do 600 i = 1, m
        f(i) = obs(itype,i)-calc(maxcal,i) + offset(itype)
 600  continue
      return
      end

**********************************************************************
      SUBROUTINE AMOVE (X1,X2,X3,X4,NPC)
c
c purpose : To set up varaibles and arrays for subroutine "DFS001"
c           so that points can be moved automatically to achieve a best
c           fit of the calculated anomaly with the observed anomaly.
c
c parameters : x1   - first independent variable,"x" for point #1
c              x2   - second    "          "     ,"z"  "    "   #1
c              x3   - third     "          "     ,"x"  "    "   #2
c              x4   - fourth    "          "     ,"z"  "    "   #2
c             npc   - number of independent variables, 2 for 1 point
c

$include:'magrav.cmn'

      data esc/100./, iwrite/0/, maxit/1/

      do 100 i = 1, maxbod
        xx(i) = 0.
100   continue
      xx(1) = x1
      xx(2) = x2
      xx(3) = x3
      xx(4) = x4
      ibr = 1
      if(nfield(itype).eq.0) then
        write(*,*)'ERROR,no observed data defined!'
        call sound(15,6000)
        return
      endif
      n1 = 0
      n2 = 0
      do 200 i = 1, 4
        ib(i) = 0
        np(i) = 0
200   continue
      do 300 number = 1, 2
        call check(number,xx(1),xx(2),ibod,npt,iret)
        if(iret.eq.0) then
          ib(number) = ibod
          np(number) = npt
          n1 = 1
```

```
        endif
        if(npc.ne.2) then
          call check(number,xx(3),xx(4),ibod,npt,iret)
          if(iret.eq.0) then
            ib(number + 2) = ibod
            np(number + 2) = npt
            n2 = 1
          endif
        endif
300   continue
      ntot = n1 + n2
      if(ntot.eq.0) return
      n3 = ntot * 2
      do 400 i = 1, n3
        e(i) = .001
400   continue
      ierr = 0
      call dfs001(nfield(itype),n3,f,xx,e,esc,iwrite,maxit,w,ierr)
      if(ierr.ne.0) return
      xposf = xpos(nfield(itype),itype)
      if(xx(1).gt.xposf) xx(1) = xposf
      if(xx(1).lt.0.)xx(1) = 0.
      if(xx(2).lt.0.)xx(2) = 0.
      if(xx(2).gt.zmax)xx(2) = zmax
      if(n3.ne.2) then
        if(xx(3).gt.xposf) xx(3) = xposf
        if(xx(3).lt.0.) xx(3) = 0.
        if(xx(4).lt.0.) xx(4) = 0.
        if(xx(4).gt.zmax) xx(4) = zmax
      endif
      k = -2
      do 500 i = 1, 2
        i1 = i + i - 1
        i2 = i1 + 1
        k = k + 2
        do 520 j = 1, 2
          l = k + j
          ibl = ib(l)
          npl = np(l)
          if(ibl.ne.0) then
            x(npl,ibl) = xx(i1)
            z(npl,ibl) = xx(i2)
            ianom(ibl) = 0
            if(npl.eq.1) then
              x(npts(ibl),ibl) = x(1,ibl)
              z(npts(ibl),ibl) = z(1,ibl)
            endif
            call plbod(ibl,-1)
          endif
520     continue
500   continue
      kalk = 1
      call planom(0,-1)
      return
      end
```

```
c
c MAGRAV2 SUBROUTINE BLOCK : MS4
c Edited : June 10  , 1986 ; J. Broome
c
$nofloatcalls
c ****************************************************************
      SUBROUTINE PLOBS
c
c Purpose : To plot observed data on the screen
c
$include:'magrav.cmn'
c
      call inidis (1,-1)
      call setcol (12)
c
      if (idiff.eq.1) return
      do 100 ipt=1 , nfield(itype)
100   w(ipt) = obs(itype,ipt) + offset(itype)
c
      call movabs (xpos(1,itype),w(1))
      call polyla(xpos(1,itype),w(1),nfield(itype))
c
      return
      end
c
c ***************************************************************
      SUBROUTINE PLBOD (IBOD,ICLR)
c
c Purpose : To plot bodies on the screen
c
c parameters : ibod - body number to be plotted
c                     "0" to plot all bodies
c              iclr  - clear window to specified colour(0-255)
c                     "-1" if window is not to be cleared
c
$include:'magrav.cmn'
c
      if(mode.eq.1.and.zmax.lt.0) then
        write(*,*)'Profile length = ',xlen(itype),' km.'
10      write(*,'(a\)')' Enter depth of cross-section plot(km): '
        call sound(20,200)
        read(*,*,err=10) zmax
        zupl = 0.0
        zlpl = zmax
      endif
      call inidis (2,iclr)
      ist = ibod
      iend = ibod
      if(ibod.eq.0) then
        ist = 1
        iend = maxbod
      endif
c
      do 100 kbod = ist , iend
      if(npts(kbod).eq.0) go to 100
```

```
        call movabs (x(1,kbod),z(1,kbod))
        call polyfa(x(1,kbod),z(1,kbod),npts(kbod),kbod)
        do 110 ipt = 1 , npts(kbod)
          call setcol(255)
          call ptabs(x(ipt,kbod),z(ipt,kbod))
110     continue
100   continue
c
      return
      end
c
c ************************************************************
c
      SUBROUTINE PLANOM (JBOD,ICLR)
c
c Purpose : To plot the calculated anomaly for body "ibod"
c
c parameters : jbod - body number to be plotted("0" for all)
c              iclr - colour to clear background to(0-255)
c                     "-1" if window is not to be cleared
c
$include:'magrav.cmn'
c
      if (kalk.eq.1) call calcan
      ibod = jbod
      if (jbod.eq.0) ibod = maxcal
      call inidis (1,iclr)
c
c plot zero line
c
      if (idiff.gt.0) then
        dum = difmax(itype) * difmin(itype)
      else
        dum = anomax(itype) * anomin(itype)
      endif
c
      if(dum.lt.0) then
        call setcol (127)
        call movabs (xlpl,0)
        call lnabs (xupl,0)
      endif
c
c plot calculated anomaly
c
      dmin = 999999.
      dmax = -999999.
      do 100 ipt=1 , nfield(itype)
      if(idiff.gt.0) then
        w(ipt)=calc(ibod,ipt)-obs(itype,ipt)-offset(itype)
        dmin = amin1(dmin,w(ipt))
        dmax = amax1(dmax,w(ipt))
      else
        w(ipt) = calc(ibod,ipt)
      endif
100   continue
```

```
c
      if(idiff.eq.1) then
        ddif = (dmax-dmin)*0.10
        difmin(itype) = dmin - ddif
        difmax(itype) = dmax + ddif
        call inidis (1,-1)
        call setcol (13)
      else
        call setcol(ibod)
      endif
c
      call movabs (xpos(1,itype),w(1))
      call polyla (xpos(1,itype),w(1),nfield(itype))
      return
      end
c
c ***********************************************************
c
      SUBROUTINE INIDIS (IWIN,ICLR)
c
c Purpose : To set the current viewport and world coordinates
c
c Parameters: iwin - "1" for anom./obse. window
c                    "2" for body window
c             iclr - "-1" don't clear window
c                    "0-255" set window to specified colour
c
$include:'magrav.cmn'
c
c Set anomaly/observed window
c
      if (iwin.eq.1) then
        call setvie ( 0.0, 0.0, 1.0, .400, -1, iclr)
c
        if (idiff.gt.0) then
          plmin = difmin(itype)
          plmax = difmax(itype)
        else
          plmin = anomin(itype)
          plmax = anomax(itype)
        endif
c
        call setwor(xlpl,plmin,xupl,plmax)
      endif
c
c Set body window
c
      if (iwin.eq.2) then
        call setvie ( 0.0, .400, 1.0, 1.0, -1, iclr)
c
        call setwor (xlpl,zlpl,xupl,zupl)
        xdis = (xupl-xlpl)/125.0
        zdis = (zlpl-zupl)/70.0
        xl = xlpl + xdis
        zl = zlpl - zdis
```

```
          x2 = xl + xdis
          z2 = zl - zdis
          call setcol (254)
          call box (xl,zl,x2,z2)
        endif
        return
        end
c
c ************************************************************
        SUBROUTINE CURPOS (IRL)
c
c Purpose : Plots cursor on screen and returns world
c            coordinates of the cursor when the button is pushed
c
c parameters : irl - set to "1" for rubberband line
c
$include:'magrav.cmn'
c
        dx = (xupl-xlpl)/10000.0
        dz = (zupl-zlpl)/7000.0
        cursx = dx * 100.0
        cursz = dz * (-150.0)
        call inithc (cursz,cursx,255)
c
10      call digit
        if(jz.gt.3000.and.jz.lt.10000) then
          xc = jx*dx + xlpl
          zc = (jz-3000)*dz + zlpl
          if(irl.eq.1) then
            call setcol (128)
            call rlnabs (xc,zc)
          else
            call movhca (xc,zc)
          endif
          if (jf.ne.4) go to 10
          call sound (10,70)
          return
        else
          go to 10
        endif
        end
c
c ************************************************************
        SUBROUTINE DIGIT
c
c purpose : To call assembly language subroutine "digpos"
c            to read the digitizer cursor position and button204i
c            status from the serial port.
c
c
$include:'magrav.cmn'
c
        call readlo (jx,jz,jf)
        if(jf.gt.127) jf = jf-128
        jz = 10000-jz
```

```
c
      return
      end
c
c ******************************************************************
c
      SUBROUTINE GRAP
c
c Purpose : To interpret digitizer pad commands and branch to
c           the desired action
c
$include:'magrav.cmn'
c
      character*30 pdriv,ans
c
      xlpl = xpos(1,itype)
      xupl = xpos(nfield(itype),itype)
      zlpl = zmax
      zupl = 0
      mode = 1
      call sav
c
      call setloc (1,1)
      call inqerr(ifun,kerr)
      if(kerr.ne.0) then
        write(*,*)'ERROR, digitizer not initialized'
        write(*,*)' run "init.bat" before MAGRAV2.'
        call sound(15,6000)
        return
      endif
      call setlat (4)
c
      go to 31000
5     call sav
10    call sound(10,1000)
      write(*,'(/a/)')' Select option on digitizer pad ...'
20    call digit
      if(jf.ne.4) go to 20
      call sound (10,70)
      izcom = jz/1000 + 1
      ixcom = jx/1000 + 1
      if(izcom.gt.3) go to 90000
      if(ixcom.gt.10) go to 90000
c
      go to (10000,20000,30000) izcom
c
c bottom row of commands
c
10000 go to(10100,10200,10300,10400,10500,10600,10700,10800,
     +10900,11000) ixcom
c
c Type body
c ---------
10100  write(*,'(/a)')' Select point in body to type.'
      call idbody(ibod,npt)
```

```
      if(ibod.gt.0)call typbod(ibod)
      go to 10
c
c Type parameters
c ---------------
10200 call typar
      go to 10
c
c Type observed
c -------------
10300 call typobs
      go to 10
c
c Type anomaly
c ------------
10400 write(*,'(/a)')' Enter body whose anomaly is to be printed'
      write(*,'(a\)')'  ("0" for combined anomaly) : '
      read(*,*,err=99000) ibod
      if(ibod.ne.0) then
        if(npts(ibod).eq.0) then
          write(*,*)'ERROR,body',ibod,' NOT defined'
          call sound(15,6000)
          go to 10
        endif
      endif
      call typano(ibod)
      go to 10
c
c Set mode to magnetics
c ---------------------
10500 if(itype.ne.2) then
        call rean
        itype = 2
        kalk = 1
        write(*,*)'You are now in MAGNETICS mode.'
        go to 31000
      else
        write(*,*)'You are already in MAGNETICS mode.'
        call sound(15,6000)
        go to 10
      endif
c
c Set mode to gravity
c -------------------
10600 if(itype.ne.1) then
        call rean
        itype = 1
        kalk = 1
        write(*,*)'You are now in GRAVITY mode.'
        go to 31000
      else
        write(*,*)'You are already in GRAVITY mode.'
        call sound(15,6000)
        go to 10
      endif
```

```
c
c Diff plot on/off
c -----------------
10700 idiff = -idiff
      if(idiff.gt.0) then
        write(*,*)'Difference mode now ON.'
        call planom (0,0)
      else
        write(*,*)'Difference mode now OFF.'
        call planom (0,0)
        call plobs
      endif
      go to 10
c
c Recalculate offset and plot observed
c -------------------------------------
10800  write(*,*)'Old offset = ',offset(itype)
       sum = 0
       do 10810 ipt=1, nfield(itype)
        sum = sum + calc(maxcal,ipt) - obs(itype,ipt)
10810  continue
      offset(itype) = sum/nfield(itype)
      write(*,*)'New offset = ',offset(itype)
      if(idiff.lt.0) then
        call plobs
      else
        call planom (0,-1)
      endif
      go to 10
c
c Set Zoom
c --------
10900 if (mode.eq.2) then
        write(*,*)'You must be in draw mode to "SET ZOOM"'
        write(*,*)' Call option "DRAW"'
        call sound(15,6000)
        go to 10
      endif
      call inidis(2,-1)
      write(*,*)'Enter L.L. zoom corner, then U.R.'
      call curpos(0)
      call delay(10)
      call delhcu
      xll = xc
      zll = zc
      dx = (xupl-xlpl)/10000.0
      dz = (zupl-zlpl)/7000.0
      dz5 = -50 * dz
10910 call digit
      if(jz.gt.3000.and.jz.lt.10000) then
        xur = jx*dx + xlpl
        zur = (jz-3000)*dz + zlpl
        if(xur.le.xll.or.zur.ge.zll) go to 10910
        if(xur.gt.xupl) xur=xupl
        call setcol(128)
```

```
      if(zur.lt.dz5) zur = zupl
        call rbox(xll,zll,xur,zur)
        if (jf.ne.4) then
          go to 10910
        else
          call sound (10,70)
          skxmin = xll
          skxmax = xur
          skzmin = zur
          skzmax = zll
          call delbox
          go to 20900
        endif
      else
        go to 10910
      endif
c
c Manual scale set
c ----------------
11000 call mscale
      mode = 1
      go to 20900
c
c Middle row of commands
c
20000 goto(20100,20200,20300,20400,20500,20600,20700,20800,20900,
     +21000) ixcom
c
c Delete body
c -----------
20100 write(*,*)' Select a point in the body to delete'
      call idbody (ibod,npt)
      if(ibod.gt.0) then
        npts(ibod) = 0
        kalk = 1
        nbods = nbods - 1
        write(*,*)'Body',ibod,' deleted'
        call plbod (0,127)
        call planom(0,-1)
        go to 5
      else
        go to 10
      endif
c
c Plot selected anomaly
c ---------------------
20200 write(*,*)'Select a unique point in the body whose anomaly'
      write(*,*)'  you wish to see'
      call idbody (ibod,npt)
      if(ibod.gt.0) call planom(ibod,-1)
      go to 10
c
20300 go to 90000
c
c Delete point
```

```
c ------------
20400 write(*,'(/a)')' Select the point to be deleted'
20410 call idbody(ibod,npt)
      if(ibod.lt.0) then
        write(*,*)'WARNING,no body points deleted.'
        go to 10
      endif
      call delete (ibod,npt,iret)
      if(iret.eq.-1) then
        call plbod(ibod,-1)
        kount = kount + 1
        write(*,*)'Point',npt,' deleted from body',ibod
        if(npts(ibod).lt.3) then
          write(*,*)'Body',ibod,' deleted(less than 3 points)'
          npts(ibod) = 0
        endif
        go to 5
      else
        write(*,*)'Point found in body',ibod,' NOT deleted.'
      go to 10
      endif
c
c Insert point
c ------------
20500 write(*,*)'Select point on one side of new point,'
      write(*,*)' then select the new point position,'
      write(*,*)' then select the other adjacent point.'
      call idbody(ibod,npt)
      if(ibod.lt.0) go to 10
      xone = xc
      zone = zc
      call sound(10,1000)
      call curpos (0)
      call delay(10)
      xnew = xc
      znew = zc
      call idbody(ibod,npt)
      if(ibod.lt.0) go to 10
      xtwo = xc
      ztwo = zc
      call check(1,xnew,znew,idupbd,iduppt,iret)
      call delhcu
      nchang = 0
      kount = 0
20510 numb = 0
20511 numb = numb + 1
      call check(numb,xone,zone,ibod,npt,iret)
      if (iret) 20560,20520,20560
20520 num2 = 0
20522 num2 = num2 + 1
20530 call check(num2,xtwo,ztwo,ibod2,npt2,iret)
      if (iret) 20511,20540,20511
20540 if(npt2.ne.(npt + 1)) go to 20522
      if (ibod.ne.ibod2) then
        write(*,*)'ERROR,2 points entered from different bodies'
```

```
          call sound(15,6000)
          go to 10
        endif
c
        call insert(xnew,znew,ibod,npt,iret)
        if (iret.eq.-1) then
          call setcol(255)
          call plbod (ibod,-1)
          write(*,*)'Point inserted in body',ibod
          nchang = nchang + 1
        else
          write(*,*)'Point not inserted'
          call sound(15,6000)
        endif
20560 if(kount.eq.0) then
          kount = 1
          saver = xone
          xone = xtwo
          xtwo = saver
          saver = zone
          zone = ztwo
          ztwo = saver
          go to 20510
        endif
        if(nchang.eq.0) then
          write(*,*)'ERROR, point not inserted'
          call sound(15,6000)
          go to 10
        endif
        go to 5
c
c Print screen
c ------------
20600 pdriv = '"halo.prn"'
c        call setprn (pdriv)
c        call chkerr
c        call gprint
c        call chkerr
        go to 10
c
20700  go to 90000
c
c Optimize contrast and calculate and plot anomaly
c ----------------------------------------------------
20800 call ampl
        go to 10
c
c Sketch mode ( draw area specified by zoom coordinated )
c-------------
20900 if(mode.ne.2) then
          if(skxmin.eq.skxmax) then
            write(*,'(/a/)')' Zoom not specified, call option "ZOOM"'
            go to 10
          endif
          mode = 2
```

```
          xlpl = skxmin
          xupl = skxmax
          zlpl = skzmax
          zupl = skzmin
          call planom (0,0)
          call plobs
          call plbod (0,127)
        else
          write(*,'(/a/)')' You are already in "SKETCH" mode'
          call sound(15,6000)
        endif
          go to 10
c
c Auto scaling
c ------------
21000 call ascale
      call planom (0,0)
      if(idiff.lt.0) call plobs
      go to 10
c
c top row of commands
c
30000 goto(30100,30200,30300,30400,30500,30600,30700,30800,30900,
     +31000) ixcom
c
c Enter body
c ------------
30100 call inidis(2,-1)
      call setcol (255)
      do 30105 ibod = 1 , maxbod
        if(npts(ibod).eq.0) go to 30108
30105 continue
      write(*,*)'ERROR,the maximum number of bodies already'
      write(*,*)'    exists (10), delete a body to continue.'
      call sound(15,6000)
      go to 10
c
30108 write(*,*)'Enter body points, in clockwise order,'
      write(*,*)'  closing the body to finish.'
      irl = 0
c
30110 call curpos (irl)
      numb = 1
      call delay(10)
30120 call check(numb,xc,zc,idupbd,iduppt,iret)
      if (iret.eq.0.and.npts(ibod).gt.1) then
        if(idupbd.eq.ibod.and.iduppt.eq.1) go to 30180
        numb = numb + 1
        go to 30120
      endif
      irl = 1
      if (npts(ibod).ge.maxnpt) then
        write(*,*)'Max. no. of points entered, body closed.'
        call sound(15,6000)
        go to 30180
```

```
      endif
      npts(ibod) = npts(ibod) + 1
      if(npts(ibod).eq.1) then
        call movabs(xc,zc)
        call delhcu
      else
        call delln
        call setcol (255)
        call lnabs (xc,zc)
      endif
      x(npts(ibod),ibod) = xc
      z(npts(ibod),ibod) = zc
      go to 30110
c
30180 npts(ibod) = npts(ibod) + 1
      call delln
      call setcol (255)
      call lnabs(xc,zc)
      x(npts(ibod),ibod) = x(1,ibod)
      z(npts(ibod),ibod) = z(1,ibod)
      call plbod (ibod,-1)
      nbods = nbods + 1
      write(*,*)'Body',ibod,' created with',npts(ibod),' points.'
      write(*,*)'Call option "EPAR" to define contrast for body'
      kalk = 1
      ianom(ibod) = 0
      go to 5
c
c Move body
c ---------
30200 write(*,*)'Select a point in the body to be moved and '
      write(*,*)' move it to its new location.'
      call idbody(ibod,npt)
      xold = xc
      zold = zc
      call movabs (xold,zold)
      call curpos (1)
      call delln
      call check(1,xc,zc,ibod2,npt2,iret)
      dx = xold - xc
      dz = zold - zc
      do 30250 ipt = 1 , npts(ibod)
        x(ipt,ibod) = x(ipt,ibod) - dx
        z(ipt,ibod) = z(ipt,ibod) - dz
30250 continue
      call plbod (0,127)
      ianom(ibod) = 0
      kalk = 1
      go to 5
c
c Change parameters
c -----------------
30300 write(*,'(/a)')' Select a point in the body whose parameters'
      write(*,*)'    are to be changed'
      call idbody (ibod,npt)
```

```
      if(ibod.gt.0) then
        call param (ibod)
        go to 5
      else
        go to 10
      endif
c
c Automatic point movement
c ------------------------
30400 write(*,*)'Select point to be moved automatically'
      call idbody(ibod,npt)
      if(ibod.gt.0) then
        write(*,*)' Processing ...'
        call amove(xc,zc,0.0,0.0,2)
        go to 5
      else
        go to 10
      endif
c
c Manual point movement
c ---------------------
30500 write(*,*)'Select point to be moved, then new position.'
      call idbody(ibod,npt)
      if(ibod.gt.0) then
        call movabs(xc,zc)
        call curpos(1)
        call check(1,xc,zc,ibod2,npt2,iret)
        x(npt,ibod) = xc
        z(npt,ibod) = zc
        if(npt.eq.1) then
          x(npts(ibod),ibod) = xc
          z(npts(ibod),ibod) = zc
        endif
        call delln
        call plbod(ibod,-1)
        kalk = 1
        ianom(ibod) = 0
        go to 5
      else
        go to 10
      endif
c
c Recover
c --------
30600 call reco
      go to 31000
c
c Return to text mode
c -------------------
30700 return
c
c Calculate anomaly with existing contrast and plot
c ----------------------------------------------------
30800 call planom (0,-1)
      go to 10
```

```
c
c Draw mode (full view)
c ---------------------
30900 if(mode.ne.1) then
        mode = 1
        xlpl = xpos(1,itype)
        xupl = xpos(nfield(itype),itype)
        zlpl = zmax
        zupl = 0
        call planom (0,0)
        call plobs
        call plbod (0,127)
      else
        write(*,'(/a/)')' You are in already in "DRAW" mode'
        call sound(15,6000)
      endif
      go to 10
c
c Redraw entire display
c ---------------------
31000 call planom(0,0)
      if(idiff.lt.0) call plobs
      call plbod (0,127)
      go to 10
c
c-------------------------------------------------------------
c
90000 write(*,*)'Command not recognized , try again ;'
      go to 10
c
99000 write(*,*)'Input ERROR !'
      go to 10
c
      end
c
c *************************************************************
c
      SUBROUTINE CHKERR
c
c purpose : To check "halo" error status and print out
c           the function producing the error and the error
c           type.
c
c parameters : ifun - "halo" function where error occurred
c              ierr - error type(see "halo" manual)
c
      call inqerr (ifun,ierr)
      if(ierr.ne.0) then
        write(*,*)'ERROR:',ierr,', in "Halo" function : ',ifun
        call sound(15,6000)
      endif
      return
      end
c
c
```

```
c *****************************************************************
      SUBROUTINE DELAY(N)
c
c Purpose : to add a delay to the program to eliminate
c     duplicate point entry from the graphics pad
c
      nl00 = n*100
      do 10 j = 1 , nl00
        k= j
 10   continue
c
      return
      end
c
c *****************************************************************
      SUBROUTINE IDBODY (IBOD,NPT)
c
c purpose : To allow bodies to be identified using the cursor
c           on the digitizer pad.
c
c parameters : ibod - body number of point selected
c                     "-1" if point not defined
c              npt  - point number in body
c
$include:'magrav.cmn'
c
      call sound(10,1000)
      call inidis(2,-1)
      call curpos (0)
      call delay(10)
      call check(1,xc,zc,ibod,npt,iret)
      call delhcu
      if(iret.eq.-1) then
        write(*,*)'ERROR,point not found in any body'
        call sound (15,6000)
        ibod = -1
      endif
      return
      end
c
c *****************************************************************
      SUBROUTINE MSCALE
c
c purpose : To allow manual setting of plot scaling parameters
c
$include:'magrav.cmn'
c
      write(*,*)'Current anomaly minimum :',anomin(itype)
      write(*,'(a\)')' Enter new anomaly minimum : '
      call sound(20,200)
      read(*,*,err=400) anomin(itype)
      write(*,*)'Current anomaly maximum :',anomax(itype)
      write(*,'(a\)')' Enter new anomaly minimum : '
      call sound(20,200)
      read(*,*,err=400) anomax(itype)
```

```
      if(anomax(itype).le.anomin(itype)) then
        write(*,*)'ERROR anomaly min. is larger then max.'
        call sound(15,6000)
        return
      endif
      write(*,*)'Current crossection depth(km) : ',zlpl
      write(*,'(a\)')' Enter new SKETCH crossection depth : '
      call sound(20,200)
      read(*,*,err=400) skzmax
      write(*,'(a\)')' Modify the X SKETCH limits (y/n) : '
      call sound(20,200)
      read(*,'(a)',err=400) ans
      if(ans.eq.'y'.or.ans.eq.'Y') then
        write(*,1000) skxmin,skxmax
1000    format(1h ,'Current X min and max = ',2f9.2)
        write(*,'(a\)')' Enter new SKETCH X min. : '
        call sound(20,200)
        read(*,*,err=400) skxmin
        write(*,'(a\)')' Enter new SKETCH X max. : '
        call sound(20,200)
        read(*,*,err=400) skxmax
      endif
      return
c
400   write(*,*)'Input ERROR'
      call sound(15,6000)
      return
      end
```

```
c
c Magrav.cmn - Common block for program MAGRAV2
c Edited May 10, 1986 , by J. Broome
c
      character*10 name,kommnt,ix
      integer*2 moddat(1525),modrec(4125)
      equivalence (name,moddat(1)),(name,modrec(1))
c
      common/prog/ iscope,model,iscr,itype,ibr,maxnpt,
     +  maxbod,maxobs,maxcal,irecov(20),nback,ix(20),lngix,nmod,
     +  idiff,mode,ntypes,xx(11),f(100),e(22),np(4),ib(4),
     +  jf,jx,jz,xc,zc,xupl,xlpl,zupl,zlpl,
     +  w(1275)
c
      common/mod/ name,bdip(10),nfield(2),anomax(2),anomin(2),
     +  kommnt(8),zmax,space(2),xlen(2),kalk,xoffs(2),
     +  xton,dec,dip,zcon(2),x(20,10),z(2010),npts(10),
     +  rmmin(10,2),rhomag(10,2),rmmax(10,2),bdec(10),bdy(10),
     +  obs(2,100),ianom(10),skxmin,skxmax,skzmin,skzmax,
     +  offset(2),nbods,difmin(2),difmax(2),xdis,zdis,
     +  calc(11,100),xpos(100,2)
```

APPENDIX D

Compiling and linking procedure.

In order to generate an executable "magrav2.exe" file from the source code provided, you need the following software products:

1) A Microsoft FORTRAN 77 Compiler (Version 3.20 or higher)
3) The Multi-Halo Graphics subroutine library with Microsoft FORTRAN support (Version 2.26 or higher)
4) MS-DOS (PC-DOS) operating system.(Version 2.00 or higher)

The following steps will produce an executable MAGRAV2. The batch files provided are set up for a hard disc where all files are in the same directory.

1) Compile the 5 FORTRAN source files :

a) magrav2.for
b) ms1.for
c) ms2.for
d) ms3.for
e) ms4.for

The FORTRAN source code for magrav2 is divided into 5 files because the Microsoft compiler is unable to compile all the code in one run. The batch file "mfcomp.bat" can be used to compile the source code files. The files "magrav.cmn" and compiler files "for1.exe" and "pas2.exe" must also be on the default drive. To compile "magrav2.for", enter : 'mfcomp magrav2 <cr>'
When you have run "mfcomp.bat" on all of the source code files you will have 5 object files :

a) magrav2.obj
b) ms1.obj
c) ms2.obj
d) ms3.obj
e) ms4.obj

2) Link the object files and libraries to produce "magrav2.exe"

The batch file "mlink.bat" provided to link all the files and libraries together is written for a hard disc drive where all the files are stored on the same drive. If you do not have a hard disc, the libraries can be stored on different discs which are inserted in the drive when requested by the linker program. An alternative is to edit "mlink.bat" to indicate the correct locations of the files.

The following files are used during linking and should all be in the same directory.

a) magrav2.obj
b) ms1.obj
c) ms2.obj
d) ms3.obj
e) ms4.obj
f) magrav2.lib
g) fortran.lib
h) 8087.lib (math.lib if no 8087)
i) halodvxx.obj (Halo file)
j) halof.lib (Halo file)
k) link.exe

To run the magrav2 linker simply type : 'mlink <cr>'

Alternate linking :
----

b) If you want the program to run on computers both with and without 8087 math processor chips, record 4 of "mlink.bat" should read :

```
LINK MAGRAV2+HALODVXX+MS1+MS2+MS3+MS4,MAGRAV2,NUL,FORTRAN+
MATH+MAGRAV2+HALOF
```

APPENDIX E

Program modification notes :

Changing the program for use with different hardware should be relatively easy. Most changes will be necessary because different graphics boards and digitizers are being used. Halo supplies device drivers for most popular graphics boards and digitizers which minimizes the modifications; however, some changes may be necessary due to the different capabilities of the equipment. Some of the possible trouble areas are outlined here but others probably exist that have not been considered.

1) Within subroutine GRINIT, the different colours are initialized using Halo subroutine "setcpa". The subroutine parameters are; colour index, red intensity, green intensity, and blue intensity. "Setcpa" is a board specific Halo function that is not used for boards with less than 256 simultaneously displayable colours. A different colour setting subroutine call may be required here.

The rubberband lines and boxes used for point movement and setting the zoom area depend upon "XOR"ing the lines and boxes onto the screen so that they can be non-destructively removed(See explanation in the Halo manual). The background colour for the body window is set to colour number 127 so that when rubberband lines are drawn in colour 128 the binary XOR of 127 and 128 results in 255 which is defined as white in GRINIT. This causes the rubberband lines and boxes to appear in white on the screen. If a board with different colour setting routines and palette size is used, the correct colour number relationship must be maintained to ensure that the rubberband function will work correctly.

2) The software supplied is designed for use with a Houston Instruments Hipad digitizer. Use of a different digitizer will probably require redesign of the digitizer control template. Subroutines DIGIT, CURPOS,and GRAP may require modification to maintain correct cursor position and program control as well. Program option control is achieved in subroutine GRAP by calculating variables "izcom" and "ixcom" which are used in computed "go to" statements to branch to different program options. Different digitizers may require different scaling factors for calculating "izcom" and "ixcom" form digitizer coordinates "jx" and "jz".

3) If a mouse is to be used for positioning in mode 3 of operation, subroutine GRAP will need extensive modification to allow program control. Since mice are not absolute positioning devices program control must be obtained through the use of pop up menus.

4) The program is currently set up for a two monitor system. If you try to use it on a single monitor system, the text

intended to be written on the text monitor will scroll the graphic display off the screen. The program could be modified for a single monitor system by making it redraw the graphic display after every text message.

APPENDIX F

The profile file format :

The profile file is an ASCII file formatted as follows :

Record (1)
- The number of reading on the profile(integer)

Record (2)
- The profile sample spacing in km (real)

Record (3) to (number of readings on the profile + 2)
- observed data values, entered 1 per line (real)

All data are free formatted; therefore, readings entered in integer form will automatically be converted to real.

APPENDIX G

Vertical gradient modelling subroutine MAG for converting MAGRAV2 for modelling vertical gradient data. To use add the code in CAPITALS to subroutine MAG.

```
c**********************************************************************
      SUBROUTINE MAG(IBOD)
c
c purpose : To calculate the vertical gradient anomaly for body "ibod"
c
c  sources bio computer note 66-1-c april 1966
c   program mag written for pdp-11 by d.heffler;agc,bio 19
c   cdc3150 fortran program mag2new,agc,bio,197...
c
c  modified for 2.5 dimensional bodies by Franca Lindia
c  using equations published by Shuey and Pasquale(1973)
c  in the journal "Geophysics".
c
c  Modified  by  John  Broome  June,1986  to  calculate  vertical
c   gradient  anomalies(MODIFICATIONS IN  CAPITAL  LETTERS).  The
c   vertical gradient anomaly is given in gammas/metre.
c
$include:'magrav.cmn'

      complex zi,zil,zi2,yi,fnl,fn2,fn,yrlc,qpx,qpz,qxsm,qzsm,czero
      complex rsum,x2lzi
c
      write(*,*)'Calculating mag. an. for body',ibod
      nf = nfield(itype)
      if(rhomag(ibod,2).eq.0.) return
      cdipd = degcos(dip)
      sdipd = degsin(dip)
      sdd =   degcos(xton-dec)
      cdip =  degcos(bdip(ibod))
      sdip =  degsin(bdip(ibod))
      sd =    degcos(xton-bdec(ibod))
      cdy =   degcos(90.-(xton-bdec(ibod)))
      sdt =   degsin(xton-dec)
      cdipsd = cdip*sd
      cdpcdy = cdip*cdy
      rhobod = rhomag(ibod,2) * 2.0
      y = bdy(ibod)
      ysq = y**2
      yd = 1.0/ysq
      yi = cmplx(0.,yd)
      npt = npts(ibod)
      czero = cmplx(0.,0.)

c  check each field point
C
      NPASS = 0
C
```

```
      do 3100 k = 1, nf
C
C ADD 1 METRE TO Z CONSTANT FOR SECOND PASS
3150    NPASS = NPASS + 1
        IF(NPASS.EQ.2) ZCON(2) = ZCON(2) + .001
C
        qxsm = czero
        qzsm = czero
        rsum = czero
        x1 = x(1,ibod) - xpos(k,itype)
        z1 = z(1,ibod) + zcon(2)
        if(z1.le.0.) goto 9999
        r1 = sqrt( x1**2 + z1**2 + ysq )
        zi1 = cmplx(0.,z1)
        do 3000 j = 2, npt
          x2 = x(j,ibod) - xpos(k,itype)
          z2 = z(j,ibod) + zcon(2)
          if(z2.le.0.) goto 9999

c  if 2 points the same check the point after

          if(x1.eq.x2.and.z1.eq.z2) goto 3000
          z21 = z2 - z1
          x21 = x2 - x1
          zi = cmplx(0.,z21)
          x21zi = x21 + zi
          zi2 = cmplx(0.,z2)
          r2 = sqrt( x2**2 + z2**2 + ysq )
          fn1 = x21zi/(x1+zi1) * (1.0 + r1/y)
     +      +  yi* (x1*z21 - z1*x21)
          fn2 = x21zi / (x2+zi2) * (1.0 + r2/y)
     +      +  yi*(x2*z21 - z2*x21)

c top and bottom of log >0. since "zcon" not = 0

          if(fn1.eq.czero) goto 9999
          if(fn2.eq.czero) goto 9999
          fn = fn2/fn1
          yr1c = clog(fn)
          qpx = zi/x21zi * yr1c
          qpz =  -x21/x21zi * yr1c
          qxsm = qxsm + qpx
          qzsm = qzsm + qpz
          rsum = rsum + yr1c
          x1 = x2
          z1 = z2
          zi1 = zi2
          r1 = r2
3000    continue
        qtot = real(qxsm)
        pxtot = aimag(qxsm)
        pztot = aimag(qzsm)
        rytot = aimag(rsum)
        h =  cdipsd*pxtot + sdip*qtot
        v = cdipsd*qtot - sdip*pztot
```

```
        hy = cdpcdy*rytot
        calc(ibod,k) = (v*sdipd + (h*sdd - hy*sdt)*cdipd) * rhobod
C
C IF NPASS = 1, THEN CALCULATE THE ANOMALY 1 M HIGHER
C IF NPASS = 2, THEN CALCULATE THE DIFFERENCE BETWEEN THE TWO
C       VALUES SEPARATED BY 1 METER FOR THE VERTICAL GRADIENT
C       IN GAMMAS/METRE
C
        IF(NPASS.EQ.1) THEN
          CALTMP = CALC(IBOD,K)
          GO TO 3150
        ELSE
          CALC(IBOD,K) = CALTMP - CALC(IBOD,K)
          ZCON (2) = ZCON(2) - 0.001
          NPASS = 0
        ENDIF
C
3100  continue
      ianom(ibod) = -1
      return

c    error

9999  continue
      write(*, *) 'Body :', ibod, ' Point :', k
      write(*, *) ' Cannot be calculated with present algorithm'
      write(*, *) ' Value out of range '
      write(*, *) ' Anomaly set to zero'
      write(*, *) ' Use "GRAV" or "MAGN" command'
     +           ,' to set a larger Z constant'
      do 10000 k = 1, nf
        calc(ibod,k) = 0.0
10000 continue
      ianom(ibod) = 0
      return
      end
```