USGS-OFR-85-377

UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

AN AUTOMATIC PROGRAM FOR THE INTERPRETATION OF TWO-DIMENSIONAL GRAVITY AND MAGNETIC ANOMALIES

by

ALEXANDER WAGINI

1985

Open-File Report 85-377

Prepared in cooperation with the
Nevada Operations Office
U. S. Department of Energy
(Interagency Agreement DE-AI08-78ET44802)

Menlo Park, California

1985

Copies of this Open-File Report

may be purchased from

Open-File Services Section

Branch of Distribution

U.S. Geological Survey

Box 25425, Federal Center

Denver, Colorado 80225

PREPAYMENT IS REQUIRED

Price information will be published

in the monthly listing

"New Publications of the Geological Survey"

FOR ADDITIONAL ORDERING INFORMATION

CALL: 303 236-7476

USGS-OFR-85-377

UNITED STATES DEPARTMENT OF INTERIOR

GEOLOGICAL SURVEY

AN AUTOMATIC PROGRAM FOR THE INTERPRETATION OF TWO-DIMENSIONAL GRAVITY AND MAGNETIC ANOMALIES

by

ALEXANDER WAGINI [1]

[1] U.S. Geological Survey, Menlo Park, CA

TABLE OF CONTENTS

INTRODUCTION

This automatic inversion program for the interpretation of two-dimensional gravity and magnetic anomalies has been developed mainly in support of the U.S. Geological Survey's effort to characterize potential radioactive-waste storage sites at the Nevada Test Site, Nevada. This effort is supported by the Nevada operations office of the Department of Energy under the Nevada Nuclear Waste Storage Investigations project. Determining subsurface shapes and extensions of geologic bodies necessitates extensive modeling of magnetic and gravity data.

Geologic models for the source of magnetic or gravity anomalies are often developed by trial and error: an approximation is made to establish an initial model, the anomaly due to the model is calculated and compared with the observed anomaly, and the model is iteratively modified to improve the agreement between calculated and observed anomalies. The method presented is not a least-squares method like other methods developed during the last few years (Bhattacharyya, 1978; Corbato, 1965; Johnson, 1969; Montalvo, 1984), but minimizes the sum of the squares of the residuals by varying only one variable (coordinate) at a time. Varying one variable at a time allows one to use all available information in the model calculation, which can essentially reduce the computation time.

The objective of this program is to find the shape of geologic bodies when the physical parameters are known. Except for the outermost corners, only the z-coordinate of each corner-point is varied. The variation of only one variable at a time has the advantage that a large number of bodies and corner-points (in this program up to 50 bodies, each with up to 50 corner-points) can be used for the model calculation without solving a large

matrix. This can be important, especially for smaller computers.

Known parts of the geologic bodies can be kept fixed. Limits for the vertical and horizontal extent of each body can be entered in various ways. A partial knowledge about the boundaries of a body can be used by starting the sequence of variations with the more uncertain parts of the body and leaving only minor adjustments to the better known parts. The program prevents a body from extending above the topographic surface and from overlapping onto other bodies.

An automatic iterative procedure has been added to the program Hypermag (Saltus and Blakely, 1983) in the form of several subroutines as an option to normal magnetic and gravity calculations. The program is written in ANSI Standard FORTRAN 77 and is interactive; thus it requires little knowledge of the computer system and its editing facilities. A default answer is available for most of the questions, and a short explanatory message is available for some questions.

THEORY

As for typical trial-and-error techniques, we begin with an initial model. The anomaly C_i due to the initial model is calculated for every observation point i (i=1,...,k) and subtracted from the observed values O_i. A measure of the difference between observed and calculated anomalies is given by

$$E^{(1)} = \sum_{i=1}^{k} \left(O_i - C_i^{(1)}\right)^2 .$$

Subsequently, stepwise adjustment of the model parameters x_j (j=1,...,m) by a given modification Δx_j is performed to reduce E (see "General Discussion" for more details) and is given by

$$x_j^{(2)} = x_j^{(1)} + \Delta x_j .$$

After the first adjustment we get

$$E^{(2)} = \sum_{i=1}^{k} \left(O_i - C_i^{(2)}\right)^2 ,$$

where

$$C_i^{(2)} = C_i^{(1)} + \vartheta C_i^{(1)} ,$$

and

$$\vartheta C_i^{(1)} = \Delta x_1 \frac{\partial C_i}{\partial x_1} .$$

The n^{th} iteration is

$$E^{(n)} = \sum_{i=1}^{k} \left(O_i - C_i^{(n)}\right)^2 ,$$

where

$$C_i^{(n)} = C_i^{(n-1)} + \vartheta C_i^{(n-1)} .$$

The objective of each iteration step is to find $\vartheta C_i^{(n-1)}$ so that $E^{(n)}$ is minimized. Because of the approximation involved it may be necessary to repeat this procedure several times to complete the adjustment.

GENERAL DISCUSSION

The objective of the program is to find a set of geologic bodies that produce the observed anomalies. Physical parameters of the different bodies are assumed to be known and the shapes of the bodies are to be calculated. Typical of potential field problems, the solution is not unique; rather it depends on three factors:

(1) the choice of the physical parameters,
(2) the shapes used in the starting model, and
(3) the order in which the corner-points are varied.

(1) The method is most sensitive to the choice of physical parameters. If these are not chosen carefully, the program will be unable to reach a good fit. Moreover, a poor fit may indicate that the physical parameters were chosen incorrectly and should be modified.

(2) The choice of the starting model is almost as important as the choice of physical parameters. It should be geologically reasonable and as near as possible to the final model. Implausible starting models prevent the program from reaching a satisfactory fit. It is also important that the starting model include a sufficient number of corner-points. Some short-wavelength anomalies, for example, are caused by irregularities in the shape of a body near the surface, and sufficient corner-points are needed to adequately model such features. Except for the outermost corners, only the z-coordinate of

each corner-point is varied. The program does not add corners or move corners closer together. Additional corners increase the computation time; thus, it is desirable to start with few corner-points and add additional only if necessary.

(3) The magnitude of a magnetic anomaly is proportional to the inverse of the cube of the distance to the source. For the magnetic case, the program is especially sensitive to variations of near-surface corner-points. So it should be considered in advance the order in which the corner-points are varied. In general, the effect of the order in which corners are iterated is not as significant as the previously mentioned factors.

Figure 1 shows a cross section of a two-dimensional body, approximated as a polygon. The solid line in the upper section shows the anomaly produced by polygon $N_1O_1P_1Q_1$, the initial guess for the body. Subsequent iterations involve variations of corner P only and are shown as P_2, P_3, P_n, and P_{n+1}. Iterations continue as long as improvements occur. If the square deviation between the calculated and the observed values increases at any iteration, n+1, the previous location P_n is kept.

If the model shows no improvement by adding the vector A_p^1 to the corner-point P_1, the program will try to add the vector $-A_p^1$. If an improvement can be reached by adding $-A_p^1$, the program will go on by adding the vectors $-A_p^2$, $-A_p^3$, ... to the new corner-points in the same way as described above until no further improvement can be reached for this corner-point of the polygon.

The same procedure is invoked for every corner-point of the polygon. After all corner-points have been examined, the procedure should be applied a

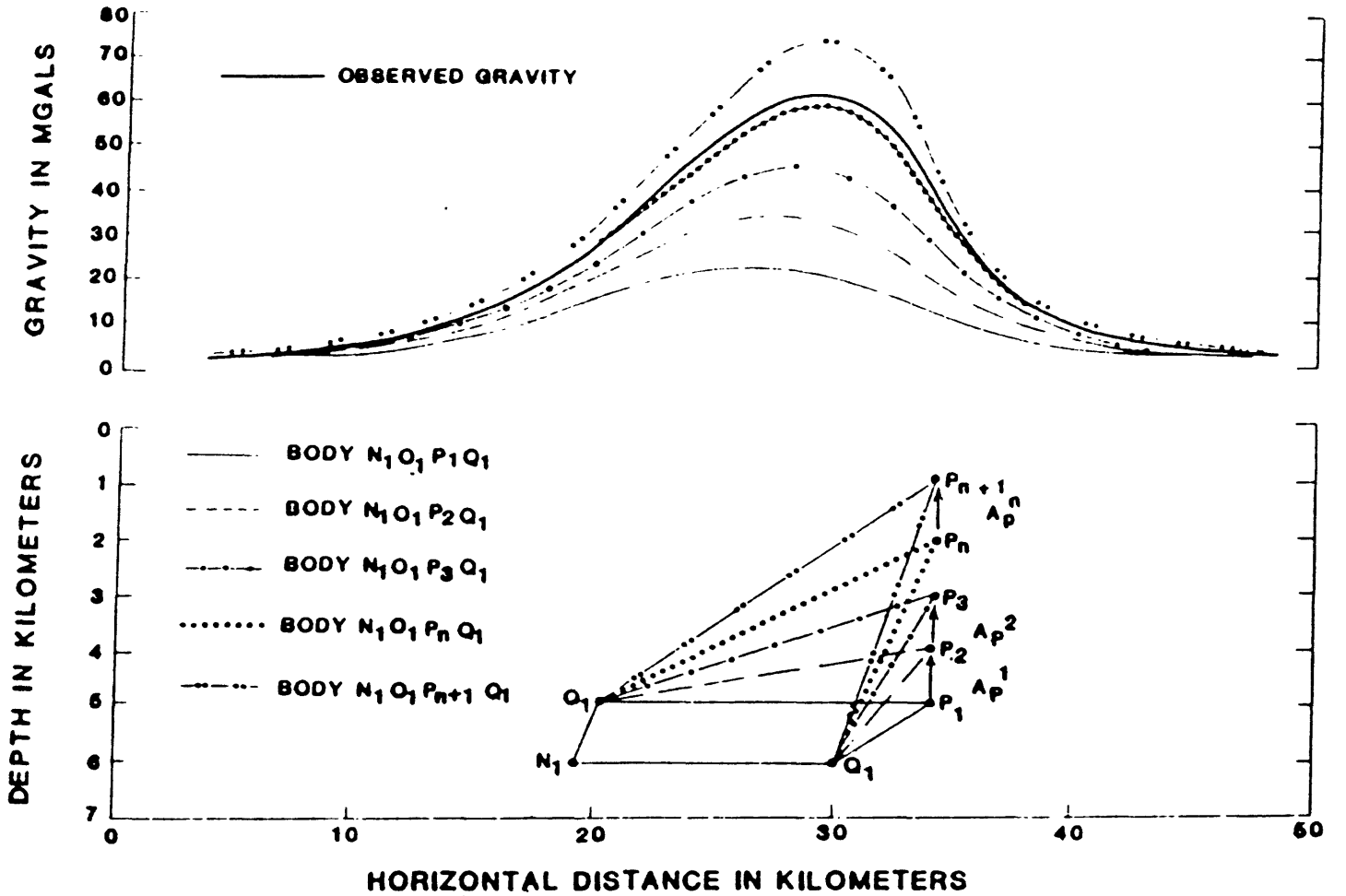

Fig. 1 - Cross section through a two-dimensional body with a density contrast of 0.3 g/cm^3. In order to improve the starting model $N_1O_1P_1Q_1$ the program will add the vectors A_p^1, A_p^2... to the corner-points P_1, P_2,... as long as the deviation between the observed and calculated values decreases. If the deviation increases by adding the vector A_p^n to the corner-point P_n, the program will stop the variation of this point.

second time with smaller vertical shifts. This method should be continued until a satisfactory fit has been reached

If two bodies have one or more corner-points in common (Fig. 2), the program assumes that the bodies are in contact with each other. When a shared corner is adjusted, both bodies are modified simultaneously.

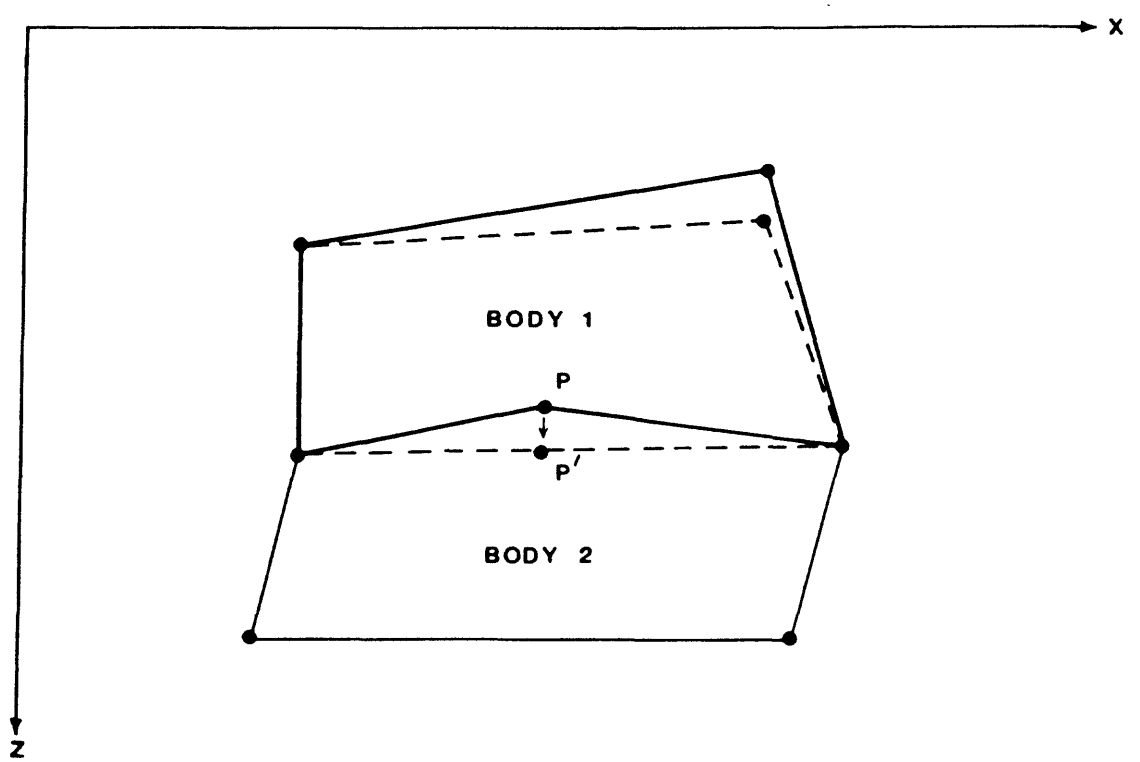

Fig. 2 - Cross section through a two-dimensional body. The solid line between body 1 and body 2 represents the contact zone of these bodies. The contact of body 1 with body 2 is maintained even if only one of the bodies has been specified for variation. In other words, if point P is varied for body 1 to P', it will automatically be changed at the same time for body 2.

INSTRUCTIONS

This section contains a step by step explanation of the dialog between the user and the program. The iterative algorithm has been implemented as an extension of program Hypermag (Saltus and Blakely, 1983); it is essential that the use of Hypermag be understood. This section only explains the automatic model calculation. The options for an answer are given in parentheses, the default answer is in brackets.

INPUT

A) Type of calculation

After you have chosen the hyper-command 'c' (HYPER-COMMAND [h]?c) to perform the model calculation, the program will offer 4 options, 2 for gravity modeling and 2 for magnetic modeling.

Q : CALCULATION TYPE (g=grav, m=mag, ag=autograv, am=automag) [m] ?

Enter 'ag' to automatically calculate the gravity effect or 'am' to calculate the magnetic effect. The default answer is 'g' or 'm' , depending on your last-calculated model.

B) Q: DO YOU WANT TO VARY BODY 1 ? (y/n) [y] :

Enter 'y' if you want to vary the coordinates of body 1. If you enter 'n' the program will ask if you want to vary the coordinates of the next body.

C) Type of variation

Q: DO YOU WANT TO VARY:

(a) ONLY THE Z-COORDINATE OF ALL POINTS ?

(b) ONLY THE X-COORDINATE OF THE OUTER POINTS ?

(c) THE Z-COORDINATE OF ALL POINTS AND THE X-COORDINATE OF THE OUTER POINTS ?

(d) ONLY SOME SPECIFIC POINTS ?

(a/b/c/d) [d] :

Enter 'a' if you want to vary only the z-coordinate of all corner-points of the body. The program assumes that the amount of each variation and the number of iterations are to be the same for each corner. The x-coordinate of all points is kept fixed.

Enter 'b' if you want to vary only the x-coordinate of the two outermost corner-points of the body. For each endpoint, the program requests the size of the variation, the number of iterations, and a horizontal limit. The z-coordinate of all points is kept fixed.

Enter 'c' if you want to vary the z-coordinate of all corner-points and the x-coordinate of the two outermost corner-points. The size of the variation and the number of iterations will be the same for all z-coordinates, but the x-coordinate of the two outermost points can be chosen individually.

Enter 'd' if you want to keep certain corner-points fixed or if you want to change the size of variation or the number of iterations for each point.

D) If 'd' is answered for question C , the program will ask if you want the z-coordinate of all corner-points to have the same size of variation and the same number of iterations.

Q: DO YOU WANT TO VARY EVERY CORNER-POINT BY THE SAME AMOUNT ? (y/n) [y] :

Enter 'n' if you want to vary the z-coordinate of some corner-points by a different amount (different size of variation) or if you want to change the number of iterations.

The default answer is 'y'.

E) Cross-control option

Certain pecularities can occur in the process of iteration which can be controlled by invoking 'cross-control'. To invoke, type 'y' in response to the following question :

Q: DO YOU WANT CROSS-CONTROL ? (h/y/n) [n] :

If you enter 'h' for help, a short explanatory message will be printed on the screen and the question will be repeated.

The default answer is 'n'.

Three things are controlled by the cross-control option:

(1) if the body extends above the topographic surface,

(2) if one body overlaps another body, and

(3) if the sides of a polygon are twisted.

(1) An iteration may cause a corner-point to move above the topographic surface. To prevent this, subroutine BERG can be invoked to report the occurrence and to move the corner-point back to the previous position. If the starting model includes a corner-point above the topographic surface, the program will report the occurrence but will not move the corner-point.

(2) A similar subroutine (SCHNITT) is invoked to determine if a corner-point lies within another body. If an iteration moves a corner-point into another body, the subroutine will report the occurrence and move the corner-point back to its previous position. If the starting model includes a corner-point within another body, the program will report the occurrence, but will not modify the corner-point.

(3) Subroutine SCHNITT is also invoked to prevent the twisting of the sides of the polygon. The program stops the variation of a corner-point at the last untwisted position.

F) Topography

It might be useful to enter topography in order to limit the extent of a body. A topographic profile can only be entered if 'cross-control' has been invoked.

Q: ARE YOUR OBSERVATION POINTS

(a) AT THE LEVEL OF THE TOPOGRAPHY ?

(b) AT A CERTAIN ALTITUDE ABOVE THE TOPOGRAPHY ?

(c) DIFFERENT FROM THE TOPOGRAPHY ?

(a/b/c) [a] :

Enter 'a' if the measurements along your profile took place on the earth's surface; i.e., the profile-points already entered represent topography.

Enter 'b' if the measurements were taken a constant altitude above the topography (e.g. an aeromagnetic drape survey). The program will then ask for the altitude.

Q: WHAT IS THE ALTITUDE ?

The answer must be given in the same units as the coordinates of the corner-points.

Enter 'c' if your observation-points are not related to the topography (e.g. a constant-level survey over an area with irregular topography). The program will then ask if topography is to be entered (see F1).

The default answer is 'a'.

F1) Topography input

Q: DO YOU WANT TO ENTER TOPOGRAPHY ?

(a) NO

(b) YES

(c) READ FROM A FILE

(a/b/c) [a] :

If you respond with 'a', the program will assume that the profile-points previously entered also describe the earth's surface; the program will check for corner-points that extend above this surface.

Enter 'b' if you want to enter topography. The program will then ask for the number of points (up to 1000) and their coordinates.

Q: HOW MANY POINTS DO YOU WANT TO ENTER ?

Q: ENTER THE X- AND Z-COORDINATES OF EACH POINT

ONE AT A TIME ...

POINT 1 : X,Z = ?

The coordinates must be in the same units as the corner-points. After the last point, the program will ask if this profile should be saved.

Q: DO YOU WANT TO SAVE YOUR TOPO-DATA ? (y/n) [y] :

If you enter 'y', you will be asked for the name of the file.

Q: NAME OF TOPO-FILE TO WRITE TO [ ] ?

The data will be written to an ASCII-file, which can be changed using the text editor independently of this program.

Enter 'c' if you want to read your topography data from an already existing file. This ASCII-file may have resulted from a previous execution of this program (subroutine WRITE_TOPO) or may have been prepared independently. This file must be formatted as follows: the first line should have a single number representing the number of points in the topographic profile; subsequent lines contain the x- and z-coordinate of each point in 'list-directed' (free) format.

Q: NAME OF THE TOPO-FILE [ ] ?

G) Vertical limit

If you have chosen the answer 'a','c', or 'd' for question C, the program will ask for the maximum and minimum vertical depths that any corner-point may have. This can be supplied in two different ways (see fig. 3):

(1) an absolute limit and
(2) relative to the topography.

(1) The program will ask for the maximum and/or minimum depth allowable. The answer should be given in the same units as the coordinates of the corner-points. The program then varies the

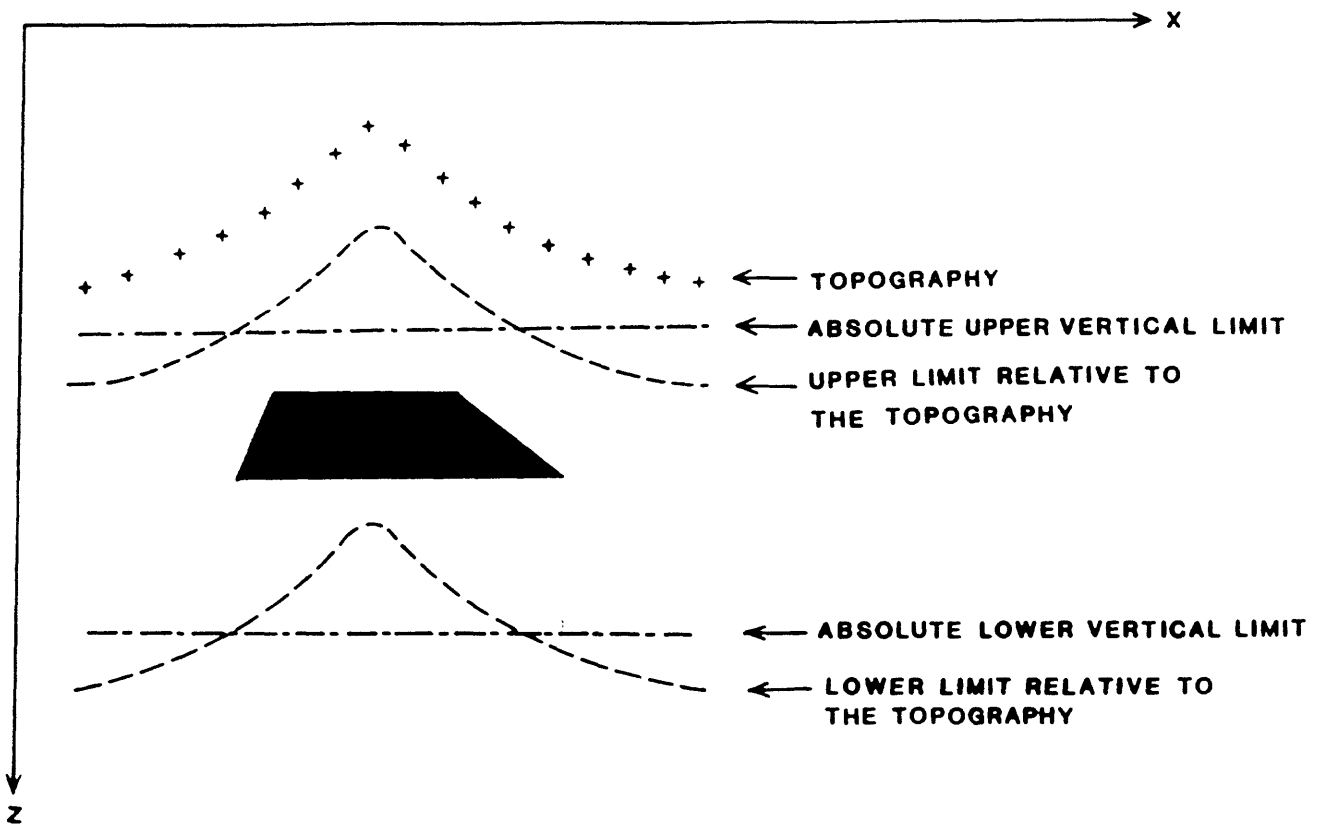

Fig. 3 - The upper and lower vertical limits for the extension of a body can be given in two different ways: as an absolute limit or relative to the topography.

corner-points of the body within these limits. If during an iteration a corner-point is adjusted to outside one of these limits, the program reports the problem and moves the point back inside. If a corner-point of the starting model is outside the limits, the program reports the occurrence but does not correct it. Separate limits may be applied to each body.

(2) If you want to limit the extent of a body relative to the topography or the surface of measurement, you can choose an upper or lower limit or both. The program automatically invokes cross-control and varies the depth of the corner-points as described above.

Q: DO YOU WANT TO GIVE A VERTICAL LIMIT TO THE BODY VARIATION ?

(a) NO

(b) RELATIVE TO THE TOPOGRAPHY

(c) AS AN ABSOLUTE LIMIT

(d) A LIMIT RELATIVE TO THE TOPOGRAPHY AND AN ABSOLUTE LIMIT

(h) HELP

(h/a/b/c/d) [a] :

Enter 'a' if no vertical limits are required.

Enter 'b' if you want to give vertical limits relative to the topography (see G2 for further information).

Enter 'c' if you want to give absolute vertical limits (see G1 for further information).

Enter 'd' if you want to give both a vertical limit relative to the topography and an absolute vertical limit. This allows you, for example, to fix the upper limit relative to the topography and the lower limit by an absolute limit. The program terminates when the first of the two limits is reached (see G1 and G2 for further information).

If you enter 'h' for help, a short explanatory message will be printed on the screen and you will be asked again to give a vertical limit.

The default answer is 'a'.

G1) If you chose to give an absolute vertical limit for the body extension (answer 'c' or 'd' in question F) the program will reply :

Q: DO YOU WANT TO FIX :
(a) THE UPPER LIMIT ?
(b) THE LOWER LIMIT ?
(c) THE UPPER AND THE LOWER LIMIT ?
(a/b/c) [c] :

Enter 'a' if you want to fix only the upper boundary by an absolute limit. The program will then ask for the limit.

Q: WHAT IS THE UPPER LIMIT ?

The answer should be entered in the same units as the coordinates of the corner-points.

Enter 'b' if you want to fix only the lower boundary by an absolute limit. The program will then ask for the limit.

Q: WHAT IS THE LOWER LIMIT ?

The answer should be entered in the same units as the coordinates of the corner-points.

Enter 'c' if you want to fix the upper and the lower boundary by an absolute limit. The program will then ask for both limits.

Q: WHAT IS THE UPPER LIMIT ?

Q: WHAT IS THE LOWER LIMIT ?

The answers should be entered in the same units as the coordinates of the corner-points.

The default answer is 'c'.

G2) If you chose to give a vertical limit for the body extension relative to the topography (answer 'b' or 'd' in question G) the program will reply with :

Q: DO YOU WANT TO FIX :

(a) THE LOWER

(b) THE UPPER

(c) THE UPPER AND THE LOWER

BODY EXTENSION RELATIVE TO THE TOPOGRAPHY ? (a/b/c) [c] :

Enter 'a' if you want to fix only the lower body extension relative to the topography. The program will then ask for the value of the limit.

Q : WHAT IS THE LOWER LIMIT RELATIVE TO THE TOPOGRAPHY ?

The answer should be entered in the same units as the coordinates of the corner-points.

Enter 'b' if you want to fix only the upper body extension relative to the topography. The program will then ask you for the value of the limit.

Q : WHAT IS THE UPPER LIMIT RELATIVE TO THE TOPOGRAPHY ?

The answer should be entered in the same units as the coordinates of the corner-points.

Enter 'c' if you want to fix the upper and the lower body extension relative to the topography. The program will then ask you for both limits.

Q: WHAT IS THE UPPER LIMIT RELATIVE TO THE TOPOGRAPHY ?

Q: WHAT IS THE LOWER LIMIT RELATIVE TO THE TOPOGRAPHY ?

The answers should be entered in the same units as the coordinates of the corner-points.

H) If you have chosen to vary the z-coordinate of all points, or of only some specific points, the program will print

********** Z-COORDINATE **********

on your screen.

I) If only certain points are to be varied, the program will prompt for each corner as follows:

Q: BODY 1 POINT 1 X: 0.00 Z: 0.00
DO YOU WANT TO VARY THE Z-COORDINATE ? (y/n) [y] :

Enter 'y' for each corner that is to be varied. The program then asks for the size of the variation (see J) and for the number of iterations (see K). If you have chosen the same variation (answer 'y' in question D) for all corner-points, the program will ask this question only once (at the first corner-point you want to vary).

J) Size of variation

To vary the coordinates of a corner-point the program needs to know the size of the variation steps.

Q: SIZE OF VARIATION [0.10] ?

The program determines a default value based on the limits of the body. The default value should only be used when the shape of the geologic body is completely unknown. A more appropriate value can be determined with the following test run. Take the default value for the size of the variation

and 3 for the number of iterations. The program will now try to adjust the calculated anomaly to the observed one. If hardly any corner-points have been moved, the size of the variation is probably too large. If most corner-points have moved 2 or 3 times and an asterisk appears at the end of most lines after the third iteration, the size of the variation is too small.

Generally, the better a model fits the observed data the smaller the size of the variation. It may be necessary to reduce the size of the variation as the model improves.

K) Number of iterations

The program also requires the number of iterations for each corner-point.

Q: NUMBER OF ITERATIONS (UP TO 8) ?

Values of 3 or 4 are usually satisfactory if the appropriate size of the variation is unknown. If you know the right size of the variation, a value of 5 or 6 should be satisfactory. A value of 7 or 8 should be used especially if the lower part of the body is still quite uncertain. If you enter a value of 0 the program will not change the coordinate of this corner-point.

L) If you have chosen to vary the x-coordinate of the outermost points (answer 'b','c' or 'd' in question C), the program will print

********** X-COORDINATE **********

on your screen.

M) If you have chosen to vary the x-coordinate of the outermost points (answer 'b','c' or 'd' in question C), the program asks first the x-coordinate of the right outermost point and then the x-coordinate of the left outermost point. If two corner-points have the same x-coordinate, the program will take the first one, the one with the lower corner-number.

Q: BODY 1 POINT 1 X: 0.00 Z: 0.00
DO YOU WANT TO VARY THE X-COORDINATE ? (y/n) [y] :

Enter 'n' if you want the x-coordinate of the corner-point fixed.

Enter 'y' if you want to vary the x-coordinate of the corner-point. The program asks for the size of the variation (see J), for the number of iterations (see K), and for a horizontal limit (see N).

N) The horizontal limit

If 'y' is answered above, the program will ask for a horizontal limit (see fig. 4). For the right side of the body the question would be :

Q: DO YOU WANT TO GIVE A HORIZONTAL LIMIT FOR :

(a) NO LIMIT

(b) THE MAXIMUM

(c) THE MINIMUM

(d) THE MAXIMUM AND MINIMUM

EXTENSION FOR THE RIGHT SIDE OF THE BODY ? (a/b/c/d) [a] :

For the left side of the body the question would be :

Q: DO YOU WANT TO GIVE A HORIZONTAL LIMIT FOR :

(a) NO LIMIT

(b) THE MAXIMUM

(c) THE MINIMUM

(d) THE MAXIMUM AND MINIMUM

EXTENSION FOR THE LEFT SIDE OF THE BODY ? (a/b/c/d) [a] :

Enter 'a' if you want to give no horizontal limit for the right or left side of the body extension.

Enter 'b' if you want to fix the maximum extension for the right or left side of the body. The program will then ask for the limit of the extension.

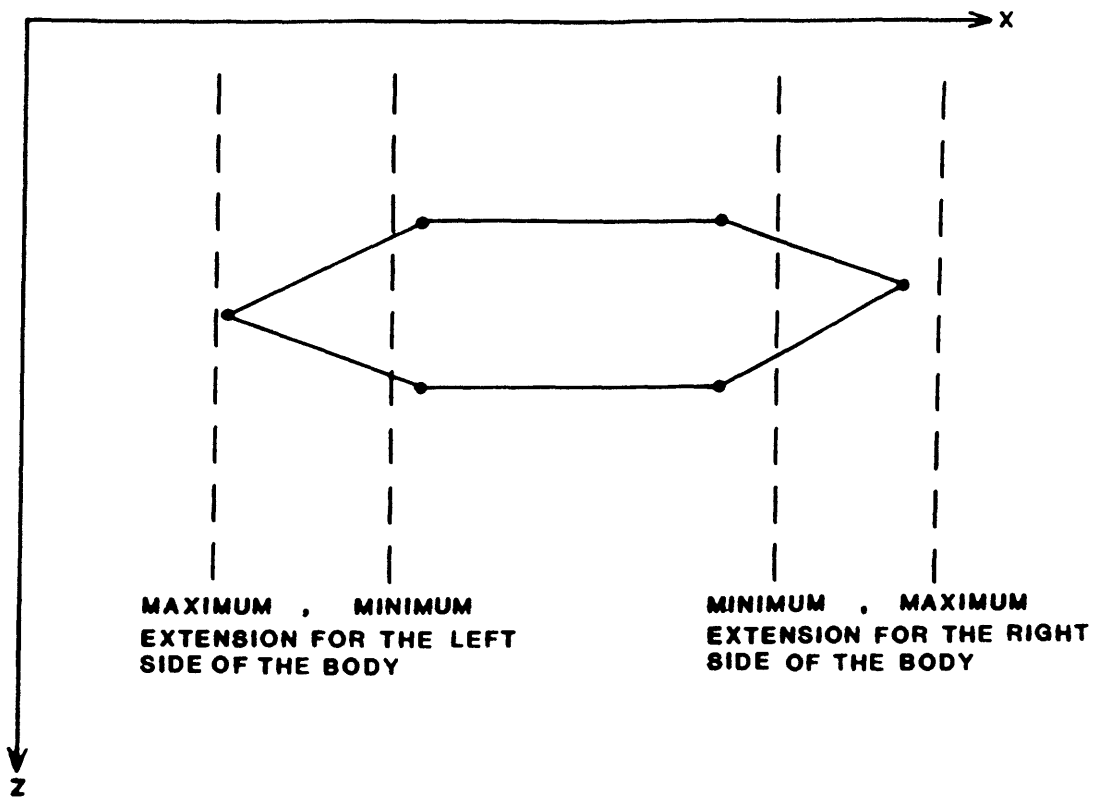

Fig. 4 - Cross section through a two-dimensional body. Two limits can be given for each of the two outermost corner-points: the minimum extension and the maximum extension.

Q: WHAT IS THE MAXIMUM EXTENSION OF THE RIGHT SIDE OF THE BODY ?

(Q: WHAT IS THE MAXIMUM EXTENSION OF THE LEFT SIDE OF THE BODY ?)

The answer should be in the same units as the coordinates of the corner-points.

Enter 'c' if you want to fix the minimum extension for the right or left side of the body. The program then asks for the limit of the extension.

Q: WHAT IS THE MINIMUM EXTENSION FOR THE RIGHT SIDE OF THE BODY ?

(Q: WHAT IS THE MINIMUM EXTENSION FOR THE LEFT SIDE OF THE BODY ?)

The answer should be in the same units as the coordinates of the corner-points.

Enter 'd' if you want to fix the minimum and the maximum extension for the right or left side of the body. The program will then ask for the limits of the extension.

Q: WHAT IS THE MAXIMUM EXTENSION FOR THE RIGHT SIDE OF THE BODY ?

Q: WHAT IS THE MINIMUM EXTENSION FOR THE RIGHT SIDE OF THE BODY ?

(Q: WHAT IS THE MAXIMUM EXTENSION FOR THE LEFT SIDE OF THE BODY ?)

(Q: WHAT IS THE MINIMUM EXTENSION FOR THE LEFT SIDE OF THE BODY ?)

The answers should be in the same units as the coordinates of the corner-points.

The default answer is 'a'.

All terminal messages due to a certain limit being exceeded are purely informational. The program automatically insures that no coordinate of a corner-point exceeds the given limits. If a message is repeated, e. g.

POINT 1 (BODY 2) HAS REACHED THE TOPOGRAPHIC SURFACE !

POINT 1 (BODY 2) HAS REACHED THE TOPOGRAPHIC SURFACE !

the starting model should be checked for the presence of corner-points outside the limits.

(A) If a corner-point of a body reaches the absolute vertical limit, one of several messages will be sent to the terminal. If absolute limits are specified, the message will be either :

POINT 1 (BODY 1) HAS REACHED THE UPPER LIMIT OF 0.0 !

or

POINT 1 (BODY 1) HAS REACHED THE LOWER limit OF 100.0 !

depending which limit has been reached.

(B) If limits are specified relative to topography and a corner-point of the body has exceeded these limits, the message will be either :

POINT 2 (BODY 3) HAS REACHED THE UPPER LIMIT !

or

POINT 2 (BODY 3) HAS REACHED THE LOWER LIMIT !

depending which limit has been reached.

(C) If horizontal limits are specified for the body extent and a corner-point of that body has exceeded these limits, the following message will appear on your terminal :

POINT 4 (BODY 2) HAS REACHED THE HORIZONTAL LIMIT OF 4000.0 !

(D) If a corner-point of one body reaches the border of another body, you will be informed by the message :

POINT 5 (BODY 4) HAS REACHED THE BORDER OF BODY 2 !

(E) If a corner-point of a body reaches the topographic surface, you will be told by the message :

POINT 5 (BODY 1) HAS REACHED THE TOPOGRAPHIC SURFACE !

(F) During the calculation the program informs you about every improvement of your model.

DEVIATION: 151.0 ITERATION: 5/6 BODY/POINT: 2/18 x: 4.6 z: 9.3

The average deviation between the observed and calculated values for each point of your profile

$$\overline{E} = \left[\frac{\sum_{i=1}^{k} (O_i - C_i)^2}{i} \right]^{\frac{1}{2}}$$

is shown by 'DEVIATION: 151.0' and decreases as the body improves; it is possible to see during the calculation how your model improves. 'ITERATION: 5/6' tells you that corner-point 18 of body 2 ('BODY/POINT: 2/18') had been moved 5 times out of 6 allowed iterations. The coordinates of the new corner-point 18 of body 2 are shown by 'x: 4.6 z: 9.3' . An asterisk at the end of this line, e.g.

DEVIATION: 151.0 ITERATION: 5/6 BODY/POINT: 2/18 x: 4.6 z: 9.3 *

occurs either because E ($E = \sum_{i=1}^{k} (O_i - C_i)^2$) decreases by less than 1 percent or because E decreases less than 50 percent of the previous iteration. In both cases the program will stop the variation of this point to prevent further unnecessary calculations.

POSSIBLE PROBLEMS

(A) Cross-control will not detect the following problems:

(1) Cross-control only checks for corner-points which extend above the topographic surface. It does not detect line segments extending above the topographic surface (Fig. 5a).

This can be corrected by adding additional corner-points to the body (Fig. 5b).

(2) Similarly, cross-control will not detect line segments that enter another polygon (Fig. 6a). This can happen when the chosen size of the variation for a corner-point of one body is bigger than the thickness of another body.

This can be corrected by choosing a smaller size of the variation (Fig. 6b).

(3) It is possible that the variation of an outer corner-point will move that point across sides of the same polygon (Fig. 7a).

This can be corrected by adding additional corner-points (Fig. 7b).

(B) The distance between the observation points determines the limit of the resolution of the model. If details of the anomaly are finer than the limit of the resolution, the fit may not be satisfactory in that area.

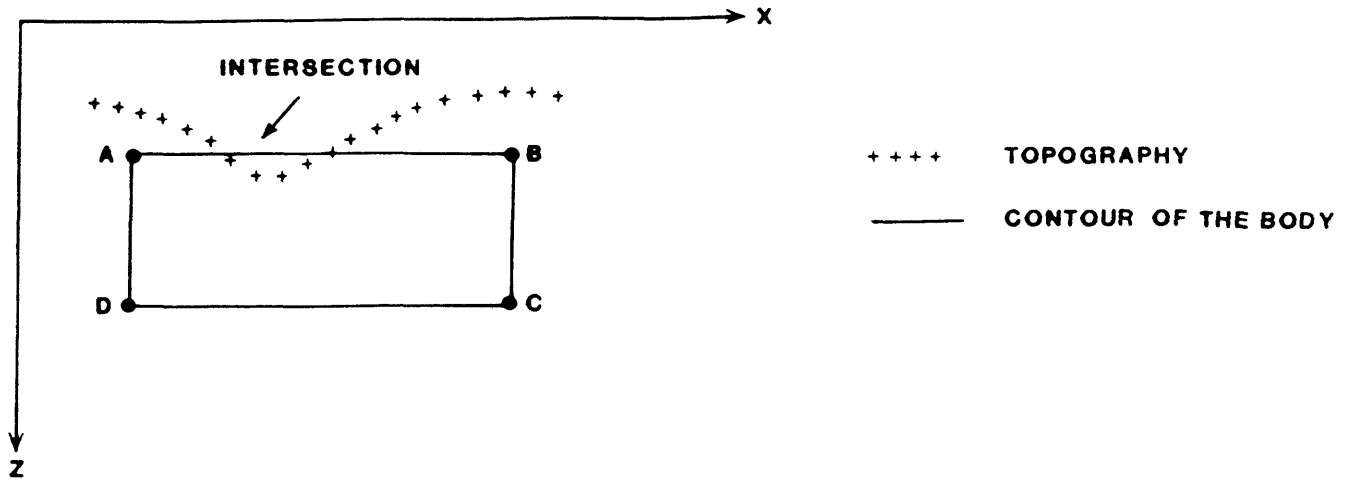

Fig. 5a - The side AB of the body ABCD intersects the Earth's surface.

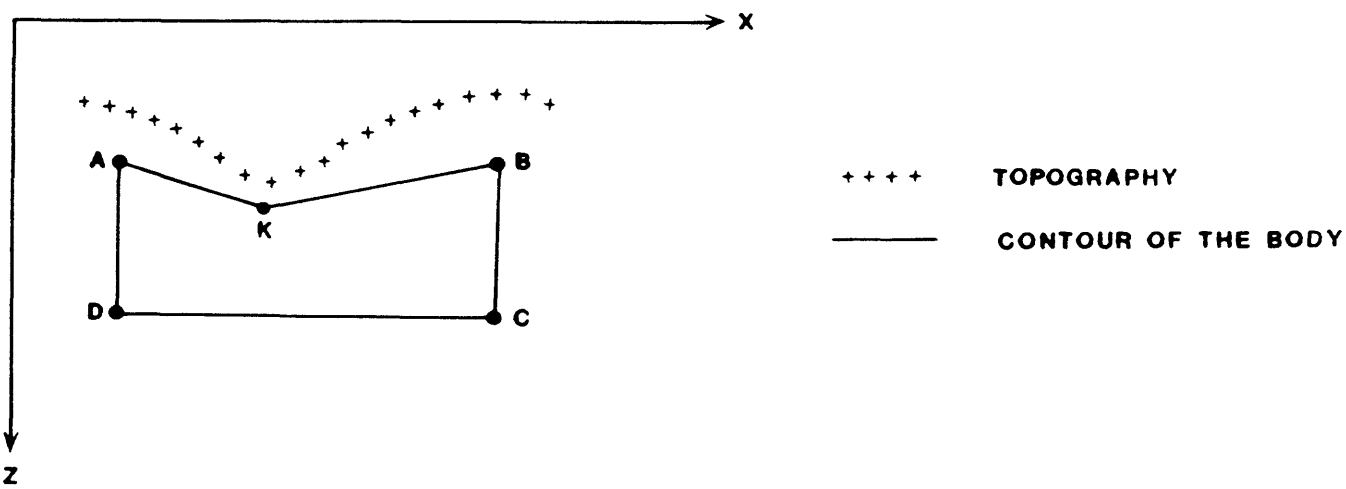

Fig. 5b - The added corner-point K prevents the intersection of the body with the Earth's surface.

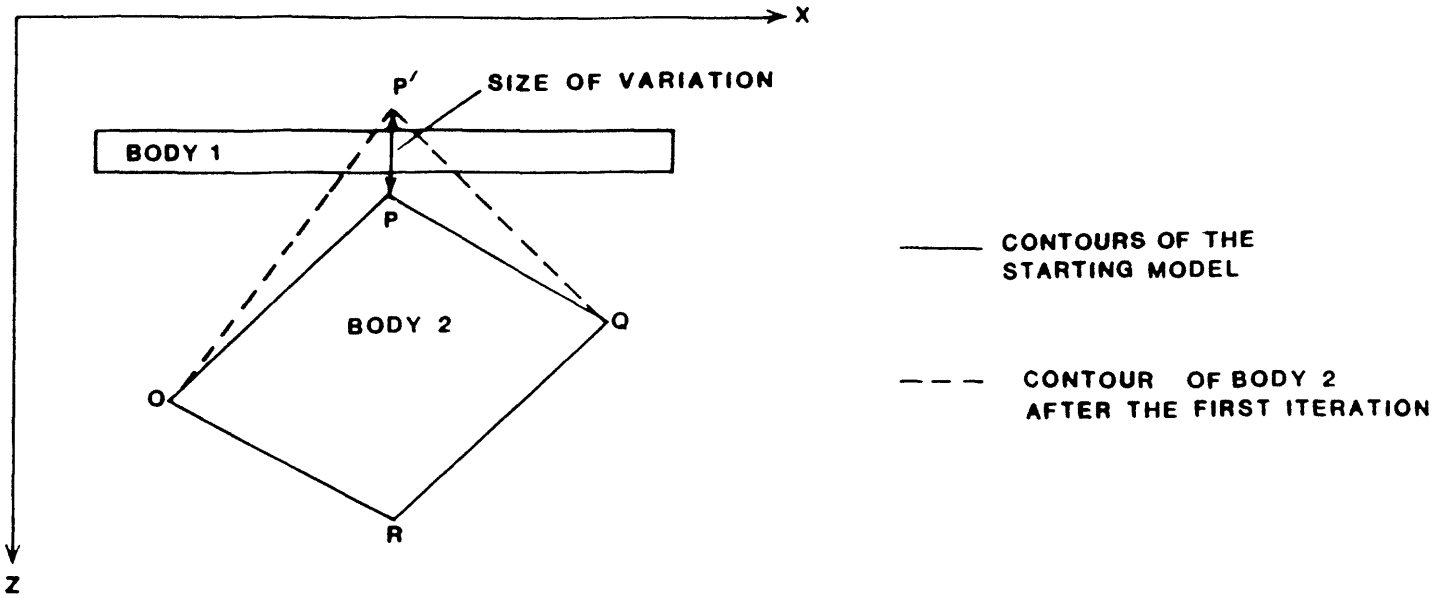

Fig. 6a - The sides OP' and P'Q of body 2 interfere with body 1, but the program will give no error message because the corner-point P' is not inside of body 2.

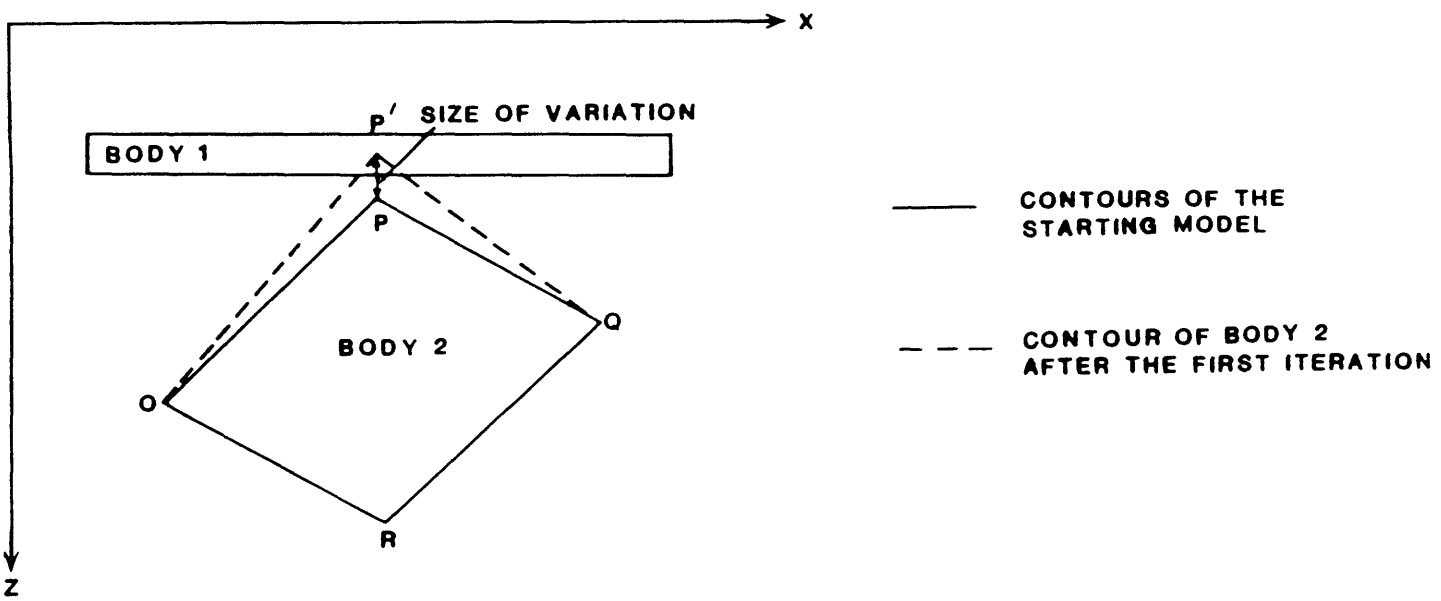

Fig. 6b - The corner-point P' is now inside of body 1, because the size of variation is now smaller than the thickness of body 2. The program will detect the error.

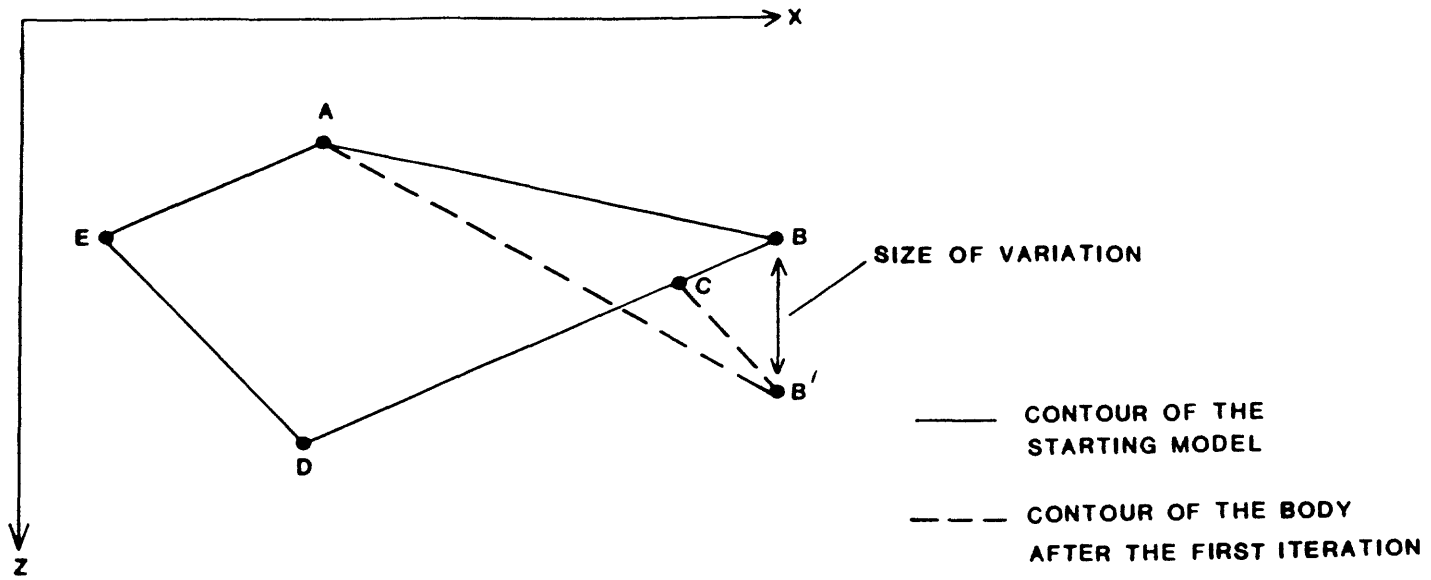

Fig. 7a - The sides AB' and CD of the body are twisted without the moving corner-point ever crossing a side. The program will not detect this error.

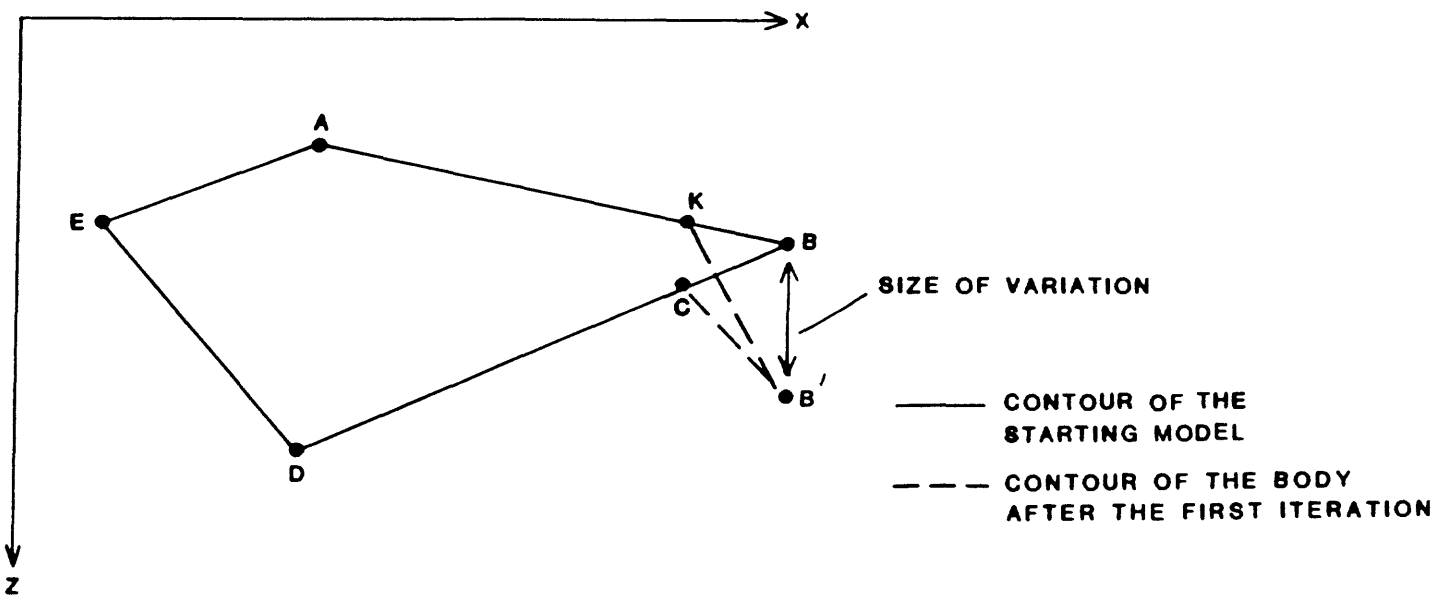

Fig. 7b - The added point K with the same x-coordinate as C prevents the twisting of the body.

For example, Figure 8 shows an anomalous observed value superimposed upon a broad anomaly. The program will calculate the best fitting curve to minimize the average square deviation for each station. In this case the observation point should be deleted.

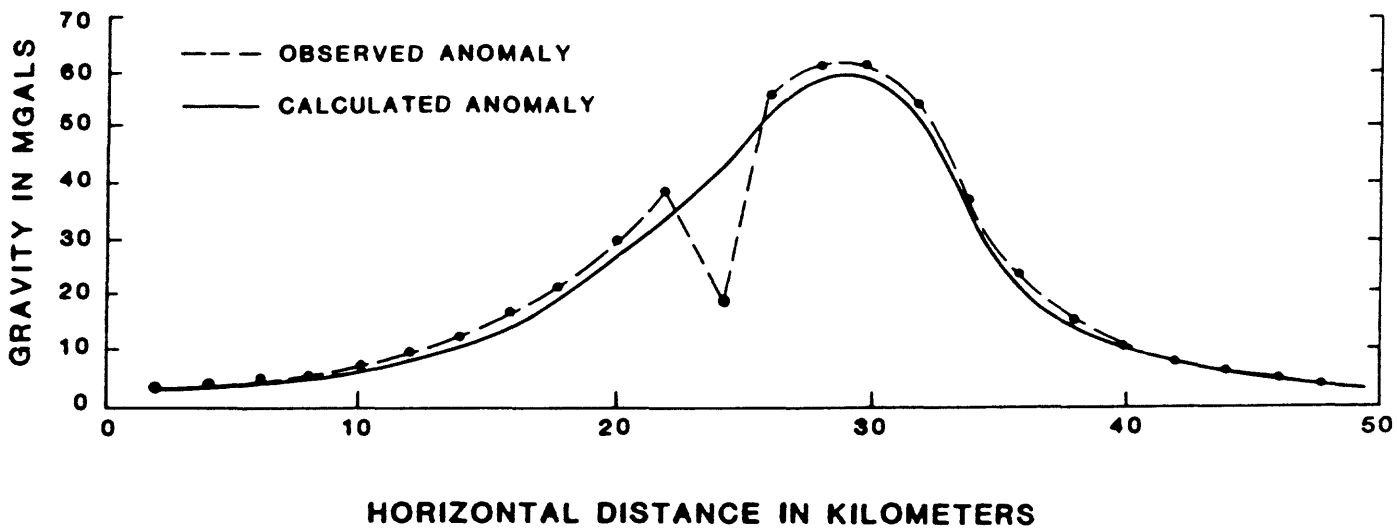

Fig. 8 - A bad observed value is superimposed upon a broader anomaly. The program will calculate the best fitting curve to bring the average square deviation for each station to a minimum.

REFERENCES

Bhattacharyya, B. K., 1978, Computer modeling in gravity and magnetic interpretation : Geophysics, v. 43, no. 5, p. 912-929.

Corbato, Ch. E., 1965, A least-square procedure for gravity interpretation : Geophysics, v. 30, no. 2, p. 228-233.

Johnson, W. W., 1969, A least-squares method of interpreting magnetic anomalies caused by two-dimensional structures : Geophysics, v.34, no. 1, p. 65-74.

Montalvo, Antonio, 1984, Automatic fitting of gravity anomalies using two-dimensional models : Geophysics, v.49, no. 3, p. 300-302.

Saltus, R. W., and Blakely, R. J., 1983, Hypermag: An interactive two-dimensional gravity and magnetic modeling program : U.S. Geological Survey Open-File Report 83-241, 28 p.

APPENDIX A : EXAMPLE 1

Magnetic case - automatic iteration for a theoretical polygonal body. The magnetization of the theoretical body (Fig. 9) and of the starting model (Fig. 10) is 4.10^{-3} emu and has the direction of the Earth's magnetic field (Inclination: 60°, Declination 14°W). The shape of the starting model is farther from the true geometry than the program would like, but it demonstrates the ability of this program.

```
Hyper-command [h]?c
Calculation type (g=grav, m=mag, ag=autograv, am=automag) [m]?am

Do you want to vary body  1 ? (y/n) [y] : y

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] : a

Do you want cross-control ? (h/y/n) [n] : y

Are your observation points
a) at the level of the topography ?
b) at a certain altitude above the topography ?
c) different from the topography ?
(a/b/c) [a] : a

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] : a

*********   Z - coordinate   *********

Size of variation [100.00] ? 800

Number of iterations (up to 8) ? 5
```

```
deviation:   445.4  iteration: 2/6  body/point: 1/ 1   x:  9000.0   z:  7200.0
deviation:   444.7  iteration: 3/6  body/point: 1/ 1   x:  9000.0   z:  6400.0*
deviation:   440.2  iteration: 2/6  body/point: 1/ 2   x: 11000.0   z:  7200.0
deviation:   435.8  iteration: 3/6  body/point: 1/ 2   x: 11000.0   z:  6400.0
deviation:   431.8  iteration: 4/6  body/point: 1/ 2   x: 11000.0   z:  5600.0
deviation:   429.1  iteration: 5/6  body/point: 1/ 2   x: 11000.0   z:  4800.0
deviation:   422.4  iteration: 2/6  body/point: 1/ 3   x: 13000.0   z:  7200.0
deviation:   415.3  iteration: 3/6  body/point: 1/ 3   x: 13000.0   z:  6400.0
deviation:   407.7  iteration: 4/6  body/point: 1/ 3   x: 13000.0   z:  5600.0
deviation:   399.9  iteration: 5/6  body/point: 1/ 3   x: 13000.0   z:  4800.0
deviation:   392.5  iteration: 6/6  body/point: 1/ 3   x: 13000.0   z:  4000.0
deviation:   384.5  iteration: 2/6  body/point: 1/ 4   x: 15000.0   z:  7200.0
deviation:   375.7  iteration: 3/6  body/point: 1/ 4   x: 15000.0   z:  6400.0
deviation:   366.3  iteration: 4/6  body/point: 1/ 4   x: 15000.0   z:  5600.0
deviation:   356.3  iteration: 5/6  body/point: 1/ 4   x: 15000.0   z:  4800.0
deviation:   346.1  iteration: 6/6  body/point: 1/ 4   x: 15000.0   z:  4000.0
deviation:   336.0  iteration: 2/6  body/point: 1/ 5   x: 17000.0   z:  7200.0
deviation:   325.5  iteration: 3/6  body/point: 1/ 5   x: 17000.0   z:  6400.0
deviation:   314.8  iteration: 4/6  body/point: 1/ 5   x: 17000.0   z:  5600.0
deviation:   304.6  iteration: 5/6  body/point: 1/ 5   x: 17000.0   z:  4800.0
deviation:   296.2  iteration: 6/6  body/point: 1/ 5   x: 17000.0   z:  4000.0
deviation:   279.2  iteration: 2/6  body/point: 1/ 6   x: 20000.0   z:  7200.0
deviation:   261.5  iteration: 3/6  body/point: 1/ 6   x: 20000.0   z:  6400.0
deviation:   244.1  iteration: 4/6  body/point: 1/ 6   x: 20000.0   z:  5600.0
deviation:   229.2  iteration: 5/6  body/point: 1/ 6   x: 20000.0   z:  4800.0
deviation:   222.3  iteration: 6/6  body/point: 1/ 6   x: 20000.0   z:  4000.0*
deviation:   200.9  iteration: 2/6  body/point: 1/ 7   x: 25000.0   z:  7200.0
deviation:   179.8  iteration: 3/6  body/point: 1/ 7   x: 25000.0   z:  6400.0
deviation:   162.3  iteration: 4/6  body/point: 1/ 7   x: 25000.0   z:  5600.0
deviation:   156.4  iteration: 5/6  body/point: 1/ 7   x: 25000.0   z:  4800.0*
deviation:   149.4  iteration: 2/6  body/point: 1/ 8   x: 30000.0   z:  7200.0
deviation:   148.2  iteration: 3/6  body/point: 1/ 8   x: 30000.0   z:  6400.0*
deviation:   148.1  iteration: 1/5  body/point: 1/ 9   x: 35000.0   z: 10800.0
deviation:   148.0  iteration: 2/5  body/point: 1/ 9   x: 35000.0   z: 11600.0*
deviation:   147.4  iteration: 1/5  body/point: 1/10   x: 30000.0   z: 11800.0
deviation:   133.9  iteration: 1/5  body/point: 1/11   x: 20000.0   z: 11800.0
deviation:   122.8  iteration: 2/5  body/point: 1/11   x: 20000.0   z: 12600.0
deviation:   114.2  iteration: 3/5  body/point: 1/11   x: 20000.0   z: 13400.0
deviation:   108.0  iteration: 4/5  body/point: 1/11   x: 20000.0   z: 14200.0
deviation:   104.1  iteration: 5/5  body/point: 1/11   x: 20000.0   z: 15000.0
deviation:    95.7  iteration: 1/5  body/point: 1/12   x: 10000.0   z: 11800.0
deviation:    89.5  iteration: 2/5  body/point: 1/12   x: 10000.0   z: 12600.0
deviation:    85.5  iteration: 3/5  body/point: 1/12   x: 10000.0   z: 13400.0
deviation:    83.5  iteration: 4/5  body/point: 1/12   x: 10000.0   z: 14200.0*
deviation:    82.9  iteration: 1/5  body/point: 1/13   x:  7000.0   z: 10800.0
deviation:    82.5  iteration: 2/5  body/point: 1/13   x:  7000.0   z: 11600.0
deviation:    82.1  iteration: 3/5  body/point: 1/13   x:  7000.0   z: 12400.0*
```

```
Results ...

body  1  point  1   x:  9000.00   z:  6400.00
body  1  point  2   x: 11000.00   z:  4800.00
body  1  point  3   x: 13000.00   z:  4000.00
body  1  point  4   x: 15000.00   z:  4000.00
body  1  point  5   x: 17000.00   z:  4000.00
body  1  point  6   x: 20000.00   z:  4000.00
body  1  point  7   x: 25000.00   z:  4800.00
body  1  point  8   x: 30000.00   z:  6400.00
body  1  point  9   x: 35000.00   z: 11600.00
body  1  point 10   x: 30000.00   z: 11800.00
body  1  point 11   x: 20000.00   z: 15000.00
body  1  point 12   x: 10000.00   z: 14200.00
body  1  point 13   x:  7000.00   z: 12400.00
```

The average deviation between the calculated and the observed total magnetic intensity for each of the 85 stations decreased during the first run of the program from about 440 nT to about 80 nT (or from 19 percent of the amplitude of the observed anomaly to 3.5 percent). Figure 11 shows the body after the first run of the program. The size of the variation for each corner-point during the first run of the program was quite large at 800 feet. A second run with a smaller size of variation, like 200 feet, will further improve the agreement between the two curves.

```
Hyper-command [h]?c
Calculation type (g=grav, m=mag, ag=autograv, am=automag) [m]?am

Do you want to vary body  1 ? (y/n) [y] :

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] : a

Do you want cross-control ? (h/y/n) [n] :

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] :
```

```
**********    Z - coordinate    **********

Size of variation [200.00] ?

Number of iterations (up to 8) ? 5



deviation:    80.9  iteration: 1/5  body/point: 1/ 1   x:  9000.0   z:  6600.0
deviation:    80.1  iteration: 2/5  body/point: 1/ 1   x:  9000.0   z:  6800.0
deviation:    79.6  iteration: 3/5  body/point: 1/ 1   x:  9000.0   z:  7000.0
deviation:    79.5  iteration: 4/5  body/point: 1/ 1   x:  9000.0   z:  7200.0*
deviation:    78.7  iteration: 1/5  body/point: 1/ 2   x: 11000.0   z:  5000.0
deviation:    76.3  iteration: 2/6  body/point: 1/ 3   x: 13000.0   z:  3800.0
deviation:    69.9  iteration: 2/6  body/point: 1/ 4   x: 15000.0   z:  3800.0
deviation:    64.4  iteration: 3/6  body/point: 1/ 4   x: 15000.0   z:  3600.0
deviation:    60.8  iteration: 4/6  body/point: 1/ 4   x: 15000.0   z:  3400.0
deviation:    60.0  iteration: 5/6  body/point: 1/ 4   x: 15000.0   z:  3200.0*
deviation:    55.7  iteration: 2/6  body/point: 1/ 5   x: 17000.0   z:  3800.0
deviation:    53.3  iteration: 3/6  body/point: 1/ 5   x: 17000.0   z:  3600.0
deviation:    53.1  iteration: 1/5  body/point: 1/ 6   x: 20000.0   z:  4200.0
deviation:    43.4  iteration: 1/5  body/point: 1/ 7   x: 25000.0   z:  5000.0
deviation:    37.0  iteration: 2/5  body/point: 1/ 7   x: 25000.0   z:  5200.0
deviation:    34.9  iteration: 3/5  body/point: 1/ 7   x: 25000.0   z:  5400.0*
deviation:    32.0  iteration: 1/5  body/point: 1/ 8   x: 30000.0   z:  6600.0
deviation:    31.3  iteration: 2/5  body/point: 1/ 8   x: 30000.0   z:  6800.0*
deviation:    31.2  iteration: 2/6  body/point: 1/ 9   x: 35000.0   z: 11400.0
deviation:    31.1  iteration: 3/6  body/point: 1/ 9   x: 35000.0   z: 11200.0*
deviation:    30.4  iteration: 1/5  body/point: 1/10   x: 30000.0   z: 12000.0
deviation:    30.2  iteration: 2/5  body/point: 1/10   x: 30000.0   z: 12200.0*
deviation:    30.0  iteration: 2/6  body/point: 1/11   x: 20000.0   z: 14800.0
deviation:    29.2  iteration: 1/5  body/point: 1/12   x: 10000.0   z: 14400.0
deviation:    28.7  iteration: 2/5  body/point: 1/12   x: 10000.0   z: 14600.0
deviation:    28.4  iteration: 3/5  body/point: 1/12   x: 10000.0   z: 14800.0*
deviation:    28.3  iteration: 2/6  body/point: 1/13   x:  7000.0   z: 12200.0
deviation:    28.2  iteration: 3/6  body/point: 1/13   x:  7000.0   z: 12000.0*

Results ...

body  1  point  1   x:  9000.00   z:  7200.00
body  1  point  2   x: 11000.00   z:  5000.00
body  1  point  3   x: 13000.00   z:  3800.00
body  1  point  4   x: 15000.00   z:  3200.00
body  1  point  5   x: 17000.00   z:  3600.00
body  1  point  6   x: 20000.00   z:  4200.00
body  1  point  7   x: 25000.00   z:  5400.00
body  1  point  8   x: 30000.00   z:  6800.00
body  1  point  9   x: 35000.00   z: 11200.00
```

The average deviation between the calculated and the observed total magnetic itensity for each station decreased during the second run of the program from about 80 nT to about 29 nT (or from 3.5 percent of the amplitude of the observed anomaly to 1.2 percent). Figure 12 shows the body after two runs with the program. The shape of the body is slightly different from the original body althought the difference between the observed and the calculated anomaly is negligible. The reason for this small difference lies in the shape of the starting model (Fig. 10). However, every body whose anomaly fits the observed values is equally valid from the point of view of potential theory, and only additional information of a different sort can help to select the right body. The program tried 96 models during these two runs. The CPU time totaled about 65 seconds on a DEC (DIGITAL EQUIPMENT CORPORATION) VAX/VMS 11 750 computer.

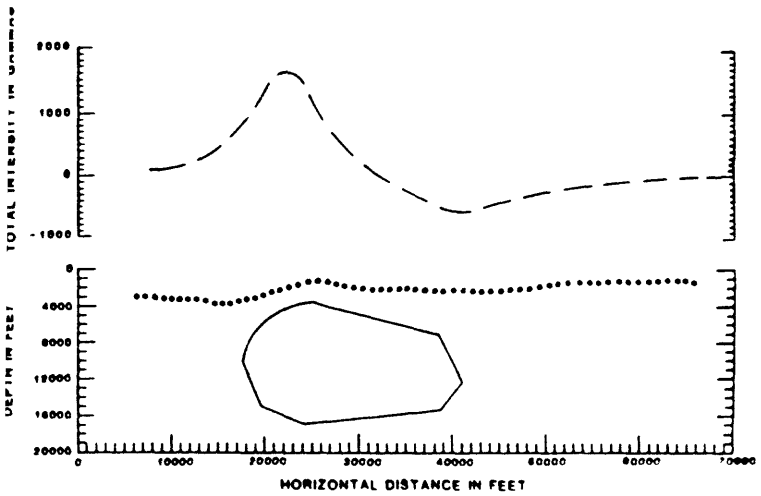

Fig. 9 - The lower portion of this figure shows a cross section through a two-dimensional, magnetic body. The outline of this theoretical body is described by a polygon. The dashed curve in the upper section shows the anomaly produced by this body.

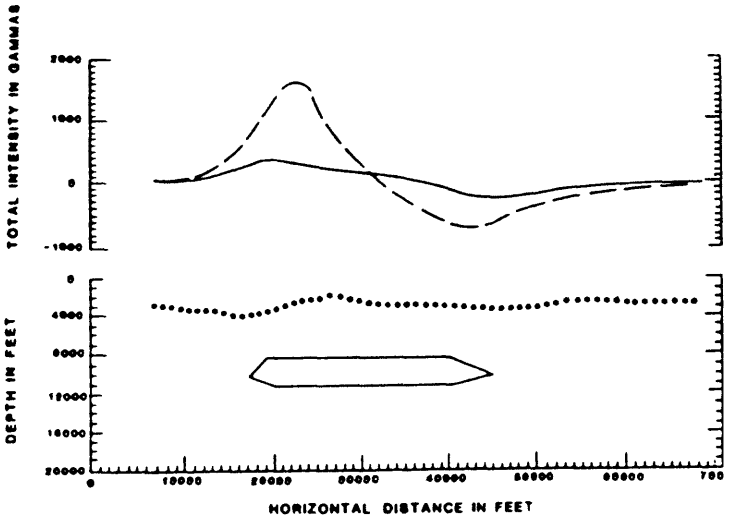

Fig. 10 - A cross section through the starting model for the calculation is shown in the lower portion of this figure. The dots represent the topography. The solid line in the upper section shows the anomaly of this body, the dashed line represents the theoretical anomaly.

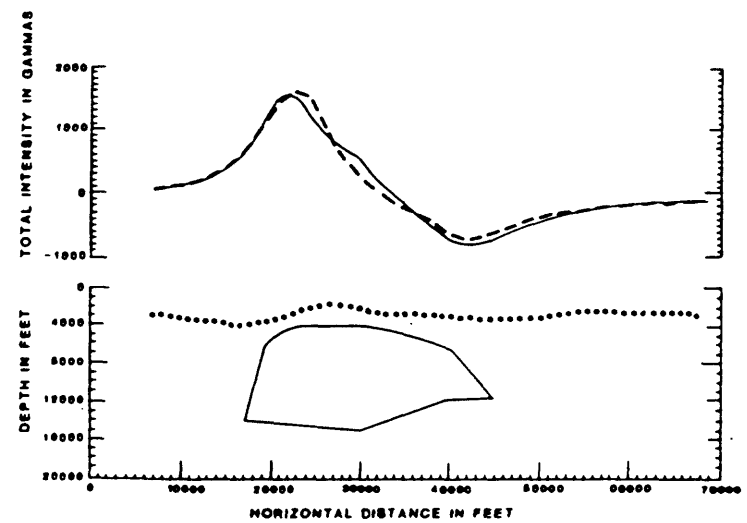

Fig. 11 - The magnetic body after the first run of the program can be seen in the lower section. The solid line in the upper section is the anomaly of this body, the dashed line again the theoretical anomaly.

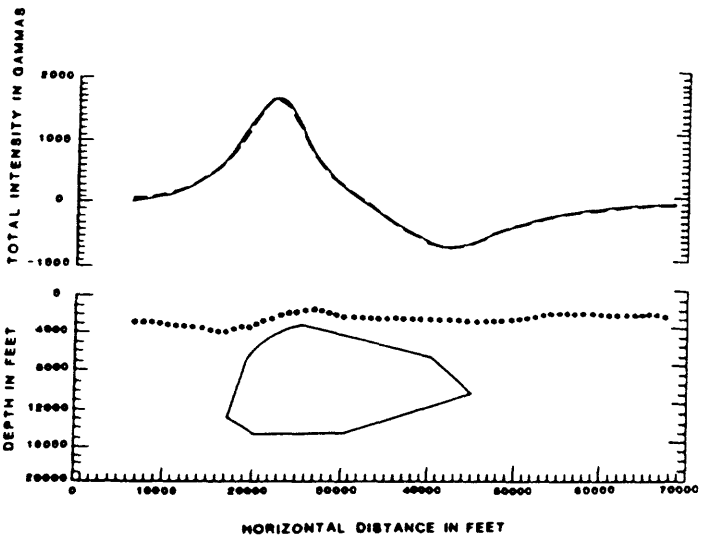

Fig. 12 - This figure shows in the lower portion the body after the second run of the program. The dashed line that is almost identical with the theoretical curve (solid line) represents the anomaly of the calculated body.

APPENDIX B : EXAMPLE 2

Gravity case - automatic iteration for a pair of theoretical bodies with different densities. The theoretical model (Fig. 13) as well as the starting model (Fig. 14) consists of two bodies with density contrasts of 0.3 g/cm^3 and 0.2 g/cm^3 respectively. Body 1 has an outcrop at the top of the hill and a contact-zone in the subsurface with body 2. Corner-points that describe the outcrop of body 1 should not be changed during the variation of the bodies, and cross-control should be invoked to prevent the two bodies from overlapping.

```
Hyper-command [h]?c
Calculation type (g=grav, m=mag, ag=autograv, am=automag) [g]?ag

Do you want to vary body  1 ? (y/n) [y] :

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] :

Do you want to vary every corner-point by  the same amount ? (y/n) [y] :

Do you want cross-control ? (h/y/n) [n] : y

Are your observation points
a) at the level of the topography ?
b) at a certain altitude above the topography ?
c) different from the topography ?
(a/b/c) [a] :

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] :
```

```
*********    Z - coordinate    *********

body  1  point  1   x: 10000.00   z:  5500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

Size of variation [100.00] ? 500.

Number of iterations (up to 8) ? 5


body  1  point  2   x: 13386.00   z:  2250.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  3   x: 14173.00   z:  2020.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  4   x: 14960.00   z:  1860.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  5   x: 15748.00   z:  1720.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  6   x: 16535.00   z:  1510.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  7   x: 17323.00   z:  1630.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  8   x: 18110.00   z:  1910.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  9   x: 18897.00   z:  2010.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 10   x: 19685.00   z:  2270.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 11   x: 20474.00   z:  2400.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 12   x: 21260.00   z:  2580.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 13   x: 22047.00   z:  2690.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 14   x: 30000.00   z:  6000.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 15   x: 25000.00   z:  6500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 16   x:  5000.00   z:  6500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 17   x:  7500.00   z:  6000.00
Do you want to vary the z-coordinate ? (y/n) [y] :
```

```
**********   X - coordinate   **********

body  1  point 14   x: 30000.00   z:  6000.00
Do you want to vary the x-coordinate ? (y/n) [y] : n

body  1  point 16   x:  5000.00   z:  6500.00
Do you want to vary the x-coordinate ? (y/n) [y] : n

Do you want to vary body  2 ? (y/n) [y] :

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] : a

Do you want cross-control ? (h/y/n) [n] : y

Are your observation points
a) at the level of the topography ?
b) at a certain altitude above the topography ?
c) different from the topography ?
(a/b/c) [a] :

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] :

**********   Z - coordinate   **********

Size of variation [100.00] ? 500.

Number of iterations (up to 8) ? 5

deviation:    2.16  iteration: 2/6  body/point: 1/ 1   x: 10000.0   z:  5000.0
deviation:    2.11  iteration: 3/6  body/point: 1/ 1   x: 10000.0   z:  4500.0
deviation:    2.08  iteration: 4/6  body/point: 1/ 1   x: 10000.0   z:  4000.0
deviation:    2.07  iteration: 5/6  body/point: 1/ 1   x: 10000.0   z:  3500.0*
deviation:    1.86  iteration: 2/6  body/point: 1/14   x: 30000.0   z:  5500.0
deviation:    1.65  iteration: 3/6  body/point: 1/14   x: 30000.0   z:  5000.0
deviation:    1.46  iteration: 4/6  body/point: 1/14   x: 30000.0   z:  4500.0
deviation:    1.31  iteration: 5/6  body/point: 1/14   x: 30000.0   z:  4000.0
deviation:    1.21  iteration: 6/6  body/point: 1/14   x: 30000.0   z:  3500.0
```

```
deviation:    1.20  iteration: 2/6  body/point: 1/17   x:  7500.0   z:  5500.0
deviation:    1.04  iteration: 2/6  body/point: 2/ 2   x: 35000.0   z:  5000.0
deviation:    0.89  iteration: 3/6  body/point: 2/ 2   x: 35000.0   z:  4500.0
deviation:    0.75  iteration: 4/6  body/point: 2/ 2   x: 35000.0   z:  4000.0
deviation:    0.68  iteration: 5/6  body/point: 2/ 2   x: 35000.0   z:  3500.0*
deviation:    0.54  iteration: 2/6  body/point: 2/ 3   x: 40000.0   z:  4750.0
deviation:    0.43  iteration: 3/6  body/point: 2/ 3   x: 40000.0   z:  4250.0
deviation:    0.42  iteration: 4/6  body/point: 2/ 3   x: 40000.0   z:  3750.0*
deviation:    0.42  iteration: 2/6  body/point: 2/ 4   x: 45000.0   z:  4500.0
deviation:    0.41  iteration: 3/6  body/point: 2/ 4   x: 45000.0   z:  4000.0
deviation:    0.41  iteration: 4/6  body/point: 2/ 4   x: 45000.0   z:  3500.0
deviation:    0.38  iteration: 2/6  body/point: 2/ 5   x: 40000.0   z:  6500.0
```

Results ...

```
body  1  point  1   x: 10000.00   z:  3500.00
body  1  point  2   x: 13386.00   z:  2250.00
body  1  point  3   x: 14173.00   z:  2020.00
body  1  point  4   x: 14960.00   z:  1860.00
body  1  point  5   x: 15748.00   z:  1720.00
body  1  point  6   x: 16535.00   z:  1510.00
body  1  point  7   x: 17323.00   z:  1630.00
body  1  point  8   x: 18110.00   z:  1910.00
body  1  point  9   x: 18897.00   z:  2010.00
body  1  point 10   x: 19685.00   z:  2270.00
body  1  point 11   x: 20474.00   z:  2400.00
body  1  point 12   x: 21260.00   z:  2580.00
body  1  point 13   x: 22047.00   z:  2690.00
body  1  point 14   x: 30000.00   z:  3500.00
body  1  point 15   x: 25000.00   z:  6500.00
body  1  point 16   x:  5000.00   z:  6500.00
body  1  point 17   x:  7500.00   z:  5500.00
body  2  point  1   x: 30000.00   z:  3500.00
body  2  point  2   x: 35000.00   z:  3500.00
body  2  point  3   x: 40000.00   z:  3750.00
body  2  point  4   x: 45000.00   z:  3500.00
body  2  point  5   x: 40000.00   z:  6500.00
body  2  point  6   x: 32500.00   z:  7000.00
body  2  point  7   x: 25000.00   z:  6500.00
```

The average deviation between the calculated and observed gravity for each of the 85 stations decreased during the first run of the program from about 2.2 mGal to about 0.38 mGal (or from 15 percent of the amplitude of the observed anomaly to 2.5 percent). Figure 14 shows the two bodies after the first run. A second run of the program with a smaller 'size of variation' will further improve the fit between the observed and calculated curves.

```
Hyper-command [q]?c
Calculation type (g=grav, m=mag, ag=autograv, am=automag) [g]?ag

Do you want to vary body  1 ? (y/n) [y] :

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] :

Do you want to vary every corner-point by  the same amount ? (y/n) [y] :

Do you want cross-control ? (h/y/n) [n] : y

Are your observation points
a) at the level of the topography ?
b) at a certain altitude above the topography ?
c) different from the topography ?
(a/b/c) [a] :

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] :

**********   Z - coordinate   **********


body  1  point  1   x: 10000.00   z:  3500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

Size of variation [100.00] ?

Number of iterations (up to 8) ? 5



body  1  point  2   x: 13386.00   z:  2250.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  3   x: 14173.00   z:  2020.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  4   x: 14960.00   z:  1860.00
Do you want to vary the z-coordinate ? (y/n) [y] : n
```

```
body  1  point  5   x: 15748.00   z:  1720.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  6   x: 16535.00   z:  1510.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  7   x: 17323.00   z:  1630.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  8   x: 18110.00   z:  1910.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point  9   x: 18897.00   z:  2010.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 10   x: 19685.00   z:  2270.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 11   x: 20474.00   z:  2400.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 12   x: 21260.00   z:  2580.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 13   x: 22047.00   z:  2690.00
Do you want to vary the z-coordinate ? (y/n) [y] : n

body  1  point 14   x: 30000.00   z:  3500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 15   x: 25000.00   z:  6500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 16   x:  5000.00   z:  6500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

body  1  point 17   x:  7500.00   z:  5500.00
Do you want to vary the z-coordinate ? (y/n) [y] :

**********   X - coordinate   **********


body  1  point 14   x: 30000.00   z:  3500.00
Do you want to vary the x-coordinate ? (y/n) [y] : n

body  1  point 16   x:  5000.00   z:  6500.00
Do you want to vary the x-coordinate ? (y/n) [y] : n
```

```
Do you want to vary body  2 ? (y/n) [y] :

Do you want to vary :
a) only the z-coordinate of all points ?
b) only the x-coordinate of the outer points ?
c) the z-coordinate of all points and the x-coordinate of the outer points ?
d) only some specific points ?
(a/b/c/d) [d] : a

Do you want cross-control ? (h/y/n) [n] : y

Are your observation points
a) at the level of the topography ?
b) at a certain altitude above the topography ?
c) different from the topography ?
(a/b/c) [a] :

Do you want to give a vertical limit to the body variation ?
a) no
b) relative  to the topography
c) as an absolute limit
d) a limit  relative to the topography and an absolute limit
h) help
(h/a/b/c/d) [a] :

*********  Z - coordinate  *********


Size of variation [100.00] ?

Number of iterations (up to 8) ? 5



deviation:    0.34  iteration: 1/5  body/point: 1/ 1   x: 10000.0   z:  3600.0
deviation:    0.31  iteration: 2/5  body/point: 1/ 1   x: 10000.0   z:  3700.0
deviation:    0.30  iteration: 1/5  body/point: 1/14   x: 30000.0   z:  3600.0
deviation:    0.30  iteration: 2/5  body/point: 1/14   x: 30000.0   z:  3700.0*
deviation:    0.28  iteration: 2/6  body/point: 1/15   x: 25000.0   z:  6400.0
deviation:    0.27  iteration: 2/6  body/point: 1/16   x:  5000.0   z:  6400.0
deviation:    0.26  iteration: 2/6  body/point: 1/17   x:  7500.0   z:  5400.0
deviation:    0.25  iteration: 3/6  body/point: 1/17   x:  7500.0   z:  5300.0
deviation:    0.25  iteration: 4/6  body/point: 1/17   x:  7500.0   z:  5200.0*
deviation:    0.23  iteration: 2/6  body/point: 2/ 1   x: 30000.0   z:  3600.0
deviation:    0.23  iteration: 3/6  body/point: 2/ 1   x: 30000.0   z:  3500.0*
deviation:    0.22  iteration: 1/5  body/point: 2/ 2   x: 35000.0   z:  3600.0
deviation:    0.21  iteration: 2/6  body/point: 2/ 3   x: 40000.0   z:  3650.0
deviation:    0.20  iteration: 3/6  body/point: 2/ 3   x: 40000.0   z:  3550.0*
deviation:    0.20  iteration: 2/6  body/point: 2/ 4   x: 45000.0   z:  3400.0
deviation:    0.20  iteration: 3/6  body/point: 2/ 4   x: 45000.0   z:  3300.0*
deviation:    0.20  iteration: 2/6  body/point: 2/ 6   x: 32500.0   z:  6900.0
```

Results ...

```
body  1  point  1   x: 10000.00   z:  3700.00
body  1  point  2   x: 13386.00   z:  2250.00
body  1  point  3   x: 14173.00   z:  2020.00
body  1  point  4   x: 14960.00   z:  1860.00
body  1  point  5   x: 15748.00   z:  1720.00
body  1  point  6   x: 16535.00   z:  1510.00
body  1  point  7   x: 17323.00   z:  1630.00
body  1  point  8   x: 18110.00   z:  1910.00
body  1  point  9   x: 18897.00   z:  2010.00
body  1  point 10   x: 19685.00   z:  2270.00
body  1  point 11   x: 20474.00   z:  2400.00
body  1  point 12   x: 21260.00   z:  2580.00
body  1  point 13   x: 22047.00   z:  2690.00
body  1  point 14   x: 30000.00   z:  3500.00
body  1  point 15   x: 25000.00   z:  6400.00
body  1  point 16   x:  5000.00   z:  6400.00
body  1  point 17   x:  7500.00   z:  5200.00
body  2  point  1   x: 30000.00   z:  3500.00
body  2  point  2   x: 35000.00   z:  3600.00
body  2  point  3   x: 40000.00   z:  3550.00
body  2  point  4   x: 45000.00   z:  3300.00
body  2  point  5   x: 40000.00   z:  6500.00
body  2  point  6   x: 32500.00   z:  6900.00
body  2  point  7   x: 25000.00   z:  6400.00
```

For each station the average deviation between the calculated and the observed gravity decreased during the second run of the program from about 0.38 mGal to about 0.20 mGal (or from 2.5 percent of the amplitude of the observed anomaly to 1.3 percent). Figure 16 shows the two bodies after the second run of the program. The program tried 74 models during these two runs. The CPU time totaled to about 76 seconds on a DEC VAX/VMS 11 750 computer.

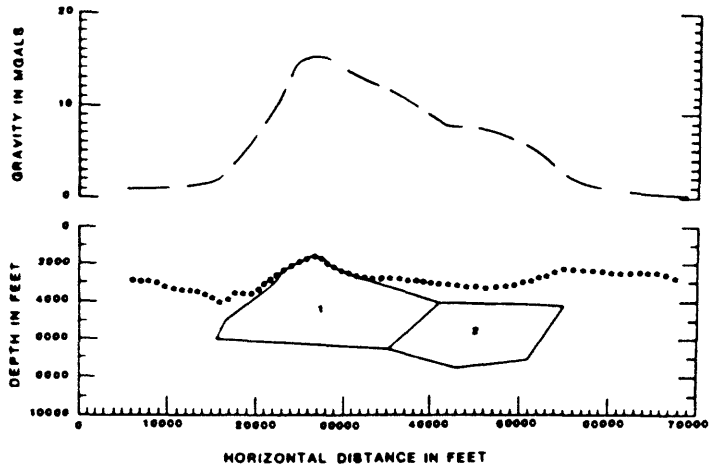

Fig. 13 - The lower portion of this figure shows a cross section through two theoretical bodies with different densities. Body 1 outcrops (the dots represent the earth's surface), and has a contact zone with body 2. The dashed line in the upper portion represents the anomaly of this theoretical model.

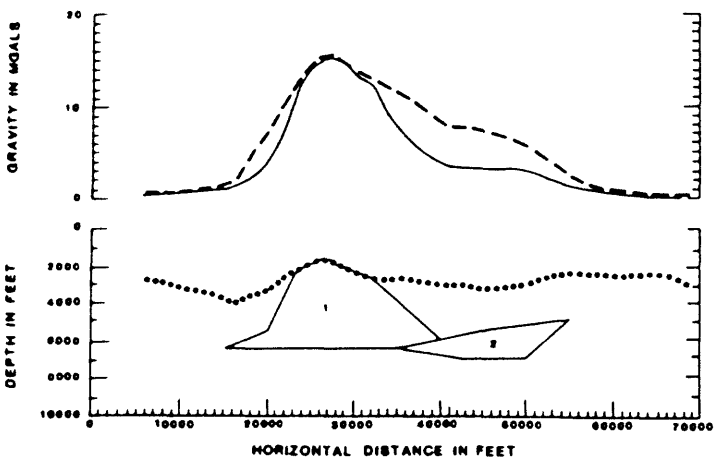

Fig. 14 - The starting model for the model calculation can be seen in the lower portion of this figure. The solid line is the anomaly of this starting model, the dashed line the theoretical anomaly.

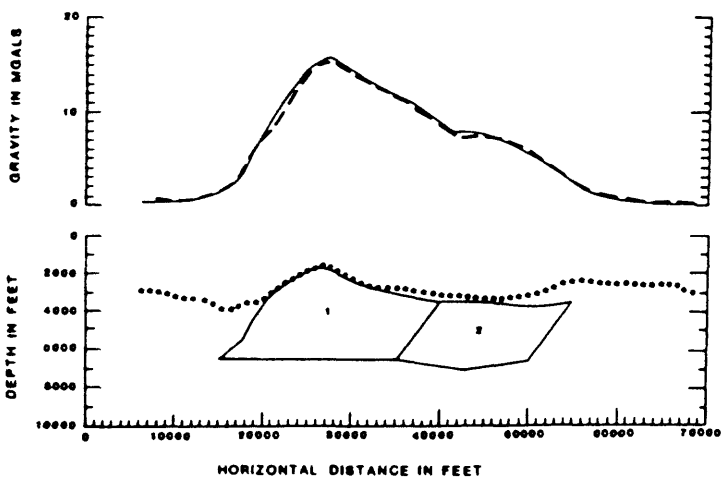

Fig. 15 - The lower portion shows the two bodies after the first run of the program. The anomaly of these bodies (solid line) fits quite well with the theoretical curve (dashed line).

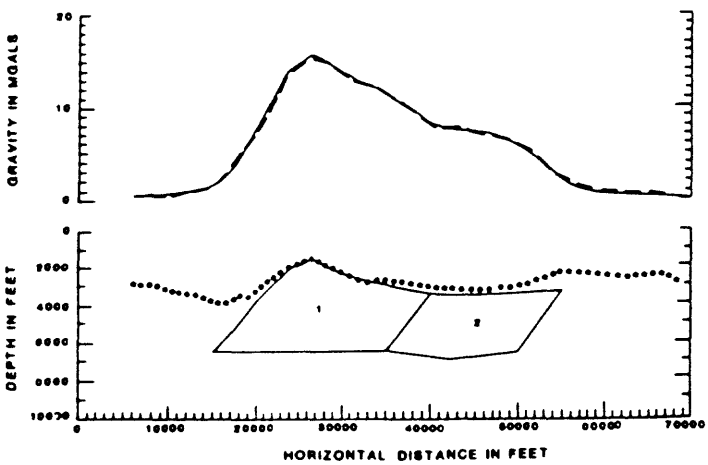

Fig. 16 - The solid line in the upper portion represents the anomaly of the two bodies shown in the lower portion. The dashed line is the theoretical anomaly.

APPENDIX C : PROGRAM LISTING

```
c.....................................................
c
c     The following subroutines have been implemented as an
c     extension of program HYPERMAG (Saltus and Blakely, 1983,
c     U.S. Geological Survey Open-File Report 83-241).
c
c     Written by: Alexander Wagini
c                 U.S. Geological Survey, MS 989
c                 345 Middlefield Road
c                 Menlo Park, CA 94025
c
c                 March 1985
c
c
c.....................................................
c
c
c
      subroutine waginim
c
c         This subroutine calls the  subroutines necessary to
c         perform the automatic iterative model calculation for
c         magnetics. The subroutine calculates and compares the
c         sum of the squares of the residuals and changes the
c         corner-points of the bodies according to the improvement
c         of the model.
c
      common /block1/ nprof,dprof,regfirst,regslope,iobs,h(1000),
     &obs(1000),x(1000),z(1000),robs(1000)
      common /block2/ nbody,xcorn(50,50),zcorn(50,50),ncorn(50),
     &m(50),mi(50),md(50),mx(50),mz(50),rho(50)
      common /block3/ azim,fi,fd,itype,xconv,zconv
      common /block7/ xmov,lfl(50),kfk(50),lrun,amin
      common /block8/ grenze(50),bod(50),jainmax(50),jainmin(50),unitmax
     1(50),grmax(50),grmin(50),itermax(50),unitmin(50),itermin(50),relgr
     2e(50)
      common /block9/ absgre(50),obergr(50),untergr(50),imaxi(50),imini(
     15),imaxex(50),iminex(50),xmaxmax(50),xmaxmin(50),xminmax(50),xmin
     2min(50)
      real m,mi,md,mx,mz
      character*1 ans,absgre*1,relgre*1,cross*1,bod*1,grenze*1
      character*1 imaxex,iminex*1,jainmax*1,jainmin*1
      dimension xc(50),zc(50),xt(1000),zt(1000),resa(1000),res(1000),
     1unit(50,50),xx(1000),zz(1000),iter(50,50),cal(1000)
c
      conv=3.1415927/180.
c
c        Body-questions
c*
      call pili(nbody,xcorn,zcorn,ncorn,cross,unit,iter,
     1xx,zz,numtopo,howalti)
c*
      fx=cos(fi*conv)*cos((fd-azim)*conv)
      fz=sin(fi*conv)
      do 3 i=1,nprof
      if(cross.eq.'y'.and.numtopo.eq.0) then
                                        xx(i)=x(i)
                                        zz(i)=z(i)+howalti
                                        endif
      xt(i)=x(i)*xconv
    3 zt(i)=z(i)*zconv
      if(numtopo.eq.0) numtopo=nprof
c
c        Body variation loop
```

```
c
      do 6 jj=1,nbody
      if(bod(jj).eq.'n') goto 6
      xmov=Ø.
c
c        Corner variation loop
c
  1Ø1 do 6 ii=Ø,ncorn(jj)
      if(xmov.eq.Ø.or.
     1xcorn(jj,ii).eq.xcorn(jj,imaxi(jj)).and.jainmax(jj).ne.'n'.and.
     2ii.eq.imaxi(jj).or.xcorn(jj,ii).eq.xcorn(jj,imini(jj)).and.
     3jainmin(jj).ne.'n'.and.ii.eq.imini(jj)) then
c
      ream=Ø.
      lrun=Ø
      if(ii.eq.Ø) go to 1Ø7
      if(iter(jj,ii).eq.Ø) goto 115
  1Ø5 lrun=lrun+1
      if(xmov.eq.Ø.) zcorn(jj,ii)=zcorn(jj,ii)+unit(jj,ii)
      if(ii.ne.imaxi(jj).or.ii.ne.imini(jj)) goto 949
      n113=Ø
c*
      call apunkt(imaxi,imini,ii,jj,ncorn,zcorn,unit,n113)
c*
      if(n113.eq.1) goto 113
  949 if(xmov.eq.1.) xcorn(jj,ii)=xcorn(jj,ii)+unit(jj,ii)
      ium=Ø
c*
      call agrenze(xmov,jj,ii,xcorn,zcorn,ium)
c*
      if(ium.eq.1) goto 113
c
c        cross-control
c
      if(cross.ne.'y') goto 1Ø7
      iuma=Ø
c*
      call kontrolle(res,xcorn,zcorn,jj,ii,grmin,grmax,
     1nbody,grenze,relgre,unit,ncorn,iuma,xx,zz,numtopo)
c*
      if(iuma.eq.1) goto 113
  1Ø7 do 8 ih=1,nprof
    8 h(ih)=Ø.
c
c        Body loop
c
      do 5 j=1,nbody
      mx(j)=cos(mi(j)*conv)*cos((md(j)-azim)*conv)
      mz(j)=sin(mi(j)*conv)
      do 4 k=1,ncorn(j)
      xc(k)=xcorn(j,k)*xconv
    4 zc(k)=zcorn(j,k)*zconv
c
c        Profile loop
c
      do 5 i=1,nprof
      if(ii.eq.Ø) resa(i)=Ø.
      res(i)=Ø.
c*
      call ribbon2(xt(i),zt(i),xc,zc,ncorn(j),mx(j),mz(j),hx,hz,ier)
c*
      if (ier.eq.Ø) go to 1Ø9
  1Ø9 h(i)=(hx*fx+hz*fz)*m(j)*1.e5+h(i)
c        calculation of the sum of the squares of the residuals
```

```
c
      if(ii.ne.0) go to 111
      cal(i)=h(i)
      resa(i)=(robs(i)-cal(i))**2+resa(i-1)
      goto 5
  111 res(i)=(robs(i)-h(i))**2+res(i-1)
    5 continue
c
c       comparison of the sum of the squares of the residuals
c
      if(ii.eq.0) amin=resa(nprof)
      if(ii.eq.0) goto 6
      if(res(nprof).ge.amin) goto 113
      iumq=0
c*
      call relbesser(lrun,amin,res,nprof,ream,iumq)
c*
      amin=res(nprof)
      amina=(amin/nprof)**0.5
      if(lrun.eq.1.and.ii.eq.1) print'(/)'
      if(iumq.eq.1) write(6,40)amina,lrun,iter(jj,ii),jj,ii,xcorn(jj,ii)
     1,zcorn(jj,ii)
   40 format(' deviation:',f8.1,'  iteration:',i2,'/',i1,'  body/point
     1:',i2,'/',i2,'   x:',f8.1,'   z:',f8.1,'*')
      if(iumq.eq.0) write(6,60)amina,lrun,iter(jj,ii),jj,ii,xcorn(jj,ii)
     1,zcorn(jj,ii)
   60 format(' deviation:',f8.1,'  iteration:',i2,'/',i1,'  body/point
     1:',i2,'/',i2,'   x:',f8.1,'   z:',f8.1)
      if(iumq.eq.1.or.lrun.ge.iter(jj,ii)) goto 115
      goto 105
  113 if(xmov.eq.0) then
                    zcorn(jj,ii)=zcorn(jj,ii)-unit(jj,ii)
      else
                    xcorn(jj,ii)=xcorn(jj,ii)-unit(jj,ii)
                    endif
      do 7 lr=1,nbody
      if(xmov.eq.0) then
         zcorn(lfl(lr),kfk(lr))=zcorn(lfl(lr),kfk(lr))-unit(jj,ii)
      else
        xcorn(lfl(lr),kfk(lr))=xcorn(lfl(lr),kfk(lr))-unit(jj,ii)
      endif
    7 continue
      if(lrun.eq.1) then
                    unit(jj,ii)=-unit(jj,ii)
                    iter(jj,ii)=iter(jj,ii)+1
                    goto 105
                    endif
  115 if(ii.ne.ncorn(jj).or.xmov.eq.1) goto 6
      unit(jj,imaxi(jj))=unitmax(jj)
      unit(jj,imini(jj))=unitmin(jj)
      iter(jj,imaxi(jj))=itermax(jj)
      iter(jj,imini(jj))=itermin(jj)
      xmov=1.
      goto 101
c
      endif
    6 continue
      print'(//,a,/)',' Results ...'
      write(6,80) ((j,i,xcorn(j,i),zcorn(j,i),i=1,ncorn(j)),j=1,nbody)
   80 format(' body ',i2,'  point ',i2,'   x:',f9.2,'   z:',f9.2)
      print'(/)'
      return
      end
c
```

```
c**********************************************************************
      subroutine pili(nbody,xcorn,zcorn,ncorn,cross,unit,iter,
     1xx,zz,numtopo,howalti)
c
c         This subroutine asks questions about the variation of
c         bodies and corner-points, about horizontal and vertical
c         limits, about the topography and about control options.
c
      common /block8/ grenze(50),bod(50),jainmax(50),jainmin(50),unitmax
     1(50),grmax(50),grmin(50),itermax(50),unitmin(50),itermin(50),relgr
     2e(50)
      common /block9/ absgre(50),obergr(50),untergr(50),imaxi(50),imini(
     150),imaxex(50),iminex(50),xmaxmax(50),xmaxmin(50),xminmax(50),xmin
     2min(50)
      dimension xcorn(50,50),zcorn(50,50),ncorn(50),unit(50,50)
     1,iter(50,50),xx(1000),zz(1000)
      character*1 zams,cross*1,bod*1,alle*1,grenze*1,absgre*1,relgre*1
      character*1 il,quest*80,iminex*1,jainmin*1,imaxex*1,jainmax*1
      character*1 protopo,jili*1,topoent*1
c
      do 1 j=1,nbody
c
c         body questions
c
      print'(/,a,i3,a,a,$)',' Do you want to vary body',j,' ? (y/n)
     1 ','[y] : '
      read'(a1)',bod(j)
      if(bod(j).eq.'n') goto 1
      grenze(j)='a'
      absgre(j)='0'
      relgre(j)='c'
      iminex(j)='a'
      imaxex(j)='a'
      jainmin(j)='n'
      jainmax(j)='n'
      itermax(j)=0
      itermin(j)=0
      unitmax(j)=0.
      unitmin(j)=0.
      xmaxmax(j)=0.
      xmaxmin(j)=0.
      xminmax(j)=0.
      xminmin(j)=0.
      grmin(j)=0.
      grmax(j)=0.
      iiiz=0
      print'(/,a,/a,/,a,/,a,/,a,/,1h$,a)',' Do you want to vary :',' a)
     1only the z-coordinate of all points ?',' b) only the x-coordinate
     2of the outer points ?',' c) the z-coordinate of all points and the
     3 x-coordinate of the outer points ?',' d) only some specific point
     4s ?','(a/b/c/d) [d] : '
      read'(a1)',alle
c*
      call xxii(j,imaxi,imini,xmaxi,xmini,zmaxi,zmini,xcorn,zcorn,ncorn)
c*
      if(alle.eq.'a'.or.alle.eq.'b'.or.alle.eq.'c') goto 119
      print'(/,1h$,a)','Do you want to vary every corner-point by
     1the same amount ? (y/n) [y] : '
      read'(a1)',zams
c
c        Cross-control
c
119   print'(/,1h$,a)','Do you want cross-control ? (h/y/n) [n] : '
      read'(a1)',cross
```

```
      if(cross.eq.'h') print'(/,a,/,a,/,a,/,a,/)',' The cross-control
     1checks :',' 1) if the body extends above the topographic
     2 surface,',' 2) if one body overlaps another body, and','
     3 3) if the sides of a polygon are twisted.'
      if(cross.eq.'h') goto 119
c
c       Topography
c
      if(cross.ne.'y') goto 125
      print'(/,a,/,a,/,a,/,a,/,1h$,a)',' Are your observation points ',
     1' a) at the level of the topography ?  ',' b) at a certain
     2 altitude above the topography ? ',' c) different from the
     3 topography ?  ','(a/b/c) [a] : '
      read'(a1)',protopo
      if(protopo.ne.'b') goto 75
      numtopo=0
      print'(/,1h$,a)','What is the altitude ? '
      read(5,100) howalti
75    if(protopo.ne.'c') goto 125
      print'(/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to enter topograp
     1hy ?',' a) no',' b) yes  ',' c) read from a file','(a/b/c) [a] : '
      read'(a1)',topoent
      if(topoent.ne.'b') goto 73
      print'(/,1h$,a)','How many points do you want to enter ? '
      read (5,*) numtopo
      print'(/,a,/,a,/)',' Enter the x- and z-coordinates of each point'
     1,' one at a time ...'
      do 5 it=1,numtopo
      print'(a,i3,a,$)',' Point ',it,' : x,z = ? '
      read (5,*) xx(it),zz(it)
5     continue
      print'(/,1h$,a)','Do you want to save your topo-data ? (y/n) [y]:'
      read'(a1)',jill
      if(jill.ne.'n') call write_topo(xx,zz,numtopo)
  73  if(topoent.eq.'c') call read_topo(xx,zz,numtopo)
c
c     Vertical limit
c
  125 if(alle.eq.'b') goto 121
  123 print'(/,a,/,a,/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to give a
     1 vertical limit to the body variation ?',' a) no',' b) relative
     2to the topography',' c) as an absolute limit',' d) a limit
     3relative to the topography and an absolute limit',
     4' h) help ','(h/a/b/c/d) [a] : '
      read'(a1)',grenze(j)
      if(grenze(j).eq.'h') print'(/,a,/,a,/,a,/,a,/,a,/)',
     1' An upper and/or lower absolute vertical limit allows the',
     2' variation of the body only within the entered depth range.',
     3' An upper and/or lower vertical limit relative to the topo
     4graphy means,',' the body can only extend to a surface para
     5llel to the topography.'
      if(grenze(j).eq.'h') goto 123
  121 if(grenze(j).ne.'b'.and.grenze(j).ne.'c'.and.grenze(j).ne.'d')
     1goto 103
c
c     absolute vertical limit
c
      if(grenze(j).eq.'b') goto 101
      print'(/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to fix :  ',' a)
     1the upper limit ? ',' b) the lower limit ?  ',' c) the up
     2per and the lower limit ?  ','(a/b/c) [c] : '
      read'(a1)',absgre(j)
      if(absgre(j).ne.'b') print'(/,1h$,a)','What is the upper
     1 limit ? '
```

```
      if(absgre(j).ne.'b') read(5,100) obergr(j)
      if(absgre(j).ne.'a') print'(/,1h$,a)','What is the lower
     1 limit ? '
      if(absgre(j).ne.'a') read(5,100) untergr(j)
c
c     relative vertical limit
c
      if(grenze(j).ne.'d') goto 103
101   cross='y'
      print'(/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to fix :  ',' a) th
     1e lower  ',' b) the upper  ',' c) the upper and the lower
     2  ','body extension relativ to the topography ? (a/b/c) [c] : '
      read'(a1)',relgre(j)
      if(relgre(j).ne.'a') print'(/,1h$,a)','What is the upper limit
     1 relative to the topography ? '
      if(relgre(j).ne.'a') read(5,100) grmin(j)
      if(relgre(j).ne.'b') print'(/,1h$,a)','What is the lower limit
     1 relative to the topography ? '
      if(relgre(j).ne.'b') read(5,100) grmax(j)
  100 format(f8.2)
c
c        Corner-questions
c
  103 do 1 i=1,ncorn(j)
      if(i.eq.1.and.alle.ne.'b') print'(/,a,/)',' **********   Z - coord
     1inate   **********'
c*
      if(i.eq.1.and.alle.ne.'b') call vargroe(zmaxi,zmini,ans)
c*
      unit(j,i)=0.
      iter(j,i)=0
      il='y'
      if(alle.eq.'b') goto 111
      if(i.eq.1.and.alle.eq.'a'.or.i.eq.1.and.alle.eq.'c') goto 105
      if(alle.eq.'a'.or.alle.eq.'c') goto 107
      write(6,140) j,i,xcorn(j,i),zcorn(j,i)
  140 format(/,' body ',i2,'  point ',i2,'   x:',f9.2,'   z:',f9.2)
      print'(1h$,a)','Do you want to vary the z-coordinate ? (y/n) [y]
     1 : '
      read'(a1)',il
160   format(i1)
      if(il.eq.'n') lilz=lilz+1
      if(il.eq.'n') goto 109
      if(zams.eq.'9') goto 107
105   unit(j,i)=ans
      print*,' '
      quest='Size of variation'
      ival=irquest(quest,unit(j,i),'(f8.2)',8)
      unita=unit(j,i)
      write(6,200)
200   format(/,1h$,'Number of iterations (up to 8) ? ')
      read(5,160) iter(j,i)
      itera=iter(j,i)
      if(zams.ne.'n') zams='9'
      print'(/)'
      goto 109
  107 unit(j,i)=unita
      iter(j,i)=itera
  109 if(alle.eq.'a') goto 1
  111 if(i.ne.ncorn(j)) goto 1


c       X-coordinate
```

```
c
      print'(/,a,/)','**********   X - coordinate   **********'
c*
      call vargroe(xmaxi,xmini,ans)
c*
c           Right outermost corner-point
c
      write(6,140) j,imaxi(j),xcorn(j,imaxi(j)),zcorn(j,imaxi(j))
      print'(1h$,a)','Do you want to vary the x-coordinate ? (y/n)
     1[y] : '
      read'(a1)',jainmax(j)
      if(jainmax(j).eq.'n') goto 115
      unitmax(j)=ans
      print*,' '
      quest='Size of variation'
      ival=irquest(quest,unitmax(j),'(f8.2)',8)
      write(6,200)
      read(5,160) itermax(j)
c
c          Horizontal limit for the right outermost corner-point
c
      print'(/,a,/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to give a horiz
     1ontal limit for :  ',' a) no limit  ',' b) the maximum  ',' c)
     2the minimum ',' d) the maximum and minimum ','extension for the
     3right side of the body ? (a/b/c/d) [a] : '
      read'(a1)',imaxex(j)
      if(imaxex(j).ne.'b'.and.imaxex(j).ne.'d') goto 113
      print'(/,1h$,a)','What is the maximum extension for the right side
     1 ? '
      read(5,100) xmaxmax(j)
  113 if(imaxex(j).ne.'c'.and.imaxex(j).ne.'d') goto 115
      print'(/,1h$,a)','What is the minimum extension for the right side
     1 ? '
      read(5,100) xmaxmin(j)
c
c        Left outermost corner-point
c
  115 write(6,140) j,imini(j),xcorn(j,imini(j)),zcorn(j,imini(j))
      print'(1h$,a)','Do you want to change the x-coordinate ? (y/n)
     1[y] : '
      read'(a1)',jainmin(j)
      if(jainmin(j).eq.'n') goto 1
      unitmin(j)=ans
      print*,' '
      quest='Size of variation'
      ival=irquest(quest,unitmin(j),'(f8.2)',8)
      write(6,200)
      read(5,160) itermin(j)
c
c          Horizontal limit for the left outermost corner-point
c
      print'(/,a,/,a,/,a,/,a,/,a,/,1h$,a)',' Do you want to give a horiz
     1ontal limit for :  ',' a) no limit  ',' b) the minimum  ',' c)
     2the maximum ',' d) the maximum and minimum ','extension for the
     3left side of the body ? (a/b/c/d) [a] : '
      read'(a1)',iminex(j)
      if(iminex(j).ne.'b'.and.iminex(j).ne.'d') goto 117
      print'(/,1h$,a)','What is the minimum extension for the left side
     1 ? '
      read(5,100) xminmax(j)
  117 if(iminex(j).ne.'c'.and.iminex(j).ne.'d') goto 1
      print'(/,1h$,a)','What is the maximum extension for the left side
     1 ? '
      read(5,100) xminmin(j)
```

```
1     continue
      return
      end
c
c*********************************************************************
      subroutine vargroe(ymaxi,ymini,ans)
c
c        This subroutine determines the default value for the 'size of variation'
c
      ymami=ymaxi-ymini
      if(ymami.ge.10000.) ans=200.
      if(ymami.ge.1000..and.ymami.lt.10000.) ans=100.
      if(ymami.ge.100..and.ymami.lt.1000.) ans=ymami/20.
      if(ymami.lt.100.) ans=ymami/10.
      return
      end
c
c*********************************************************************
      subroutine xxii(j,imaxi,imini,xmaxi,xmini,zmaxi,zmini,xcorn,zcorn,
     1ncorn)
c
c       This subroutine finds the corner index, the highest and lowest value
c         of the x-coordinates of the body and the highest and
c         lowest value of the z-coordinates
c
      dimension imaxi(50),imini(50),ncorn(50),xcorn(50,50),zcorn(50,50)
      do 1 i=1,ncorn(j)
      if(i.eq.1) then
                 imaxi(j)=1
                 xmaxi=xcorn(j,1)
                 zmaxi=zcorn(j,1)
                 imini(j)=1
                 xmini=xcorn(j,1)
                 zmini=zcorn(j,1)
      else
                 if(xcorn(j,i).gt.xmaxi) then
                                         imaxi(j)=i
                                         xmaxi=xcorn(j,i)
                                         endif
                 if(zcorn(j,i).gt.zmaxi) zmaxi=zcorn(j,i)
                 if(xcorn(j,i).lt.xmini) then
                                         imini(j)=i
                                         xmini=xcorn(j,i)
                                         endif
                 if(zcorn(j,i).lt.zmini) zmini=zcorn(j,i)
                 endif
1     continue
      return
      end
c
c*********************************************************************
      subroutine agrenze(xmov,jj,ii,xcorn,zcorn,ium)
c
c          This subroutine controls and sends a message to the terminal
c
c          1) if a corner-point of the polygon exceeds the upper
c             or lower absolute vertical limit
c
c          2) if a corner-point of the polygon exceeds the maximum
c             or minimum horizontal limit
c
      common /block9/ absgre(50),obergr(50),untergr(50),imaxi(50),imini(
     15),imaxex(50),iminex(50),xmaxmax(50),xmaxmin(50),xminmax(50),xmin
     2min(50)
```

```
      character*1 imaxex,iminex,absgre
      dimension xcorn(50,50),zcorn(50,50)
      if(absgre(jj).eq.'0'.or.xmov.eq.1.) goto 83
      if(absgre(jj).eq.'a') goto 81
      if(zcorn(jj,ii).gt.untergr(jj)) write(6,20) ii,jj,untergr(jj)
20    format(' point',i3,' (body',i2,') has reached the lower boundary
     1of',f8.1)
      if(zcorn(jj,ii).gt.untergr(jj)) goto 87
81    if(absgre(jj).eq.'b') goto 83
      if(zcorn(jj,ii).lt.obergr(jj)) write(6,40) ii,jj,obergr(jj)
40    format(' point',i3,' (body',i2,') has reached the upper boundary
     1of',f8.1)
      if(zcorn(jj,ii).lt.obergr(jj)) goto 87
83    if(xcorn(jj,ii).ne.xcorn(jj,imaxi(jj))) goto 85
      if(imaxex(jj).ne.'b'.and.imaxex(jj).ne.'d') goto 91
60    format(' point',i3,' (body',i2,') has reached the horizontal
     1limit of',f8.1)
      if(xcorn(jj,ii).gt.xmaxmax(jj)) then
                                      write(6,60) ii,jj,xmaxmax(jj)
                                      goto 87
                                      endif
91    if(imaxex(jj).ne.'c'.and.imaxex(jj).ne.'d') goto 85
      if(xcorn(jj,ii).lt.xmaxmin(jj)) then
                                      write(6,60) ii,jj,xmaxmin(jj)
                                      goto 87
                                      endif
85    if(xcorn(jj,ii).ne.xcorn(jj,imini(jj))) goto 89
      if(iminex(jj).ne.'b'.and.iminex(jj).ne.'d') goto 93
      if(xcorn(jj,ii).gt.xminmax(jj)) then
                                      write(6,60) ii,jj,xminmax(jj)
                                      goto 87
                                      endif
93    if(iminex(jj).ne.'c'.and.iminex(jj).ne.'d') goto 89
      if(xcorn(jj,ii).lt.xminmin(jj)) write(6,60) ii,jj,xminmin(jj)
      if(xcorn(jj,ii).lt.xminmin(jj)) then
                                      write(6,60) ii,jj,xminmin(jj)
                                      goto 87
                                      endif

      goto 89
  87  ium=1
  89  return
      end
c
c*********************************************************************
      subroutine kontrolle (res,xcorn,zcorn,jj,ii,grmin,grmax,
     1nbody,grenze,relgre,unit,ncorn,iuma,xx,zz,nprof)
c
c        This subroutine controls and sends a message to the terminal
c
c        1) if a body extends above the topographic surface (calling subroutine BERG),
c
c        2) if one body overlaps another body (calling subroutine SCHNITT),
c
c        3) if the sides of a polygon are twisted (calling subroutine SCHNITT), and
c
c        4) if a corner-point of the polygon exceeds the upper or lower
c           vertical limit relative to the topography (calling subroutine BERG).
c
      common /block7/ xmov,ifl(50),kfk(50),lrun,amin
      character*1 relgre,grenze*1
      dimension res(1000),iflag(50),xcorn(50,50),zcorn(50,50),
     1ncorn(50),grenze(50),grmax(50),grmin(50),relgre(50),unit(50,
     250),xx(1000),zz(1000)
c
```

```
      xco=xcorn(jj,ii)
      zco=zcorn(jj,ii)
      zzco=zco-grmin(jj)
      xx(nprof+1)=xx(nprof)+1000000.
      zz(nprof+1)=zz(nprof)+100000.
      xx(nprof+2)=xx(1)-1000000.
      zz(nprof+2)=zz(1)+100000.
      nprof2=nprof+2
      goto 103
  101 zzco=zco-grmax(jj)
      sprung=1.
c*
  103 call berg (xx,zz,nprof2,xco,zzco,ifberg)
c*
      if(sprung.eq.1) goto 105
      if(xco.ge.xx(1).and.xco.le.xx(nprof).and.ifberg.eq.0.and.grmin
     1(jj).eq.0) write(6,20) ii,jj
   20 format(' point',i3,' (body',i2,') has reached the topographic
     1 surface !')
      if(xco.ge.xx(1).and.xco.le.xx(nprof).and.ifberg.eq.0.and.grmin
     1(jj).ne.0) write(6,30) ii,jj
  30  format(' point',i3,' (body',i2,') has reached the upper limit !')
      if(xco.ge.xx(1).and.xco.le.xx(nprof).and.ifberg.eq.0) goto 109
      if((grenze(jj).eq.'b'.or.'d').and.relgre(jj).ne.'b') goto 101
      goto 107
  105 sprung=0.
      if(xco.ge.xx(1).and.xco.le.xx(nprof).and.ifberg.eq.1)
     1write(6,40) ii,jj
  40  format(' point',i3,' (body',i2,') has reached the lower limit !')
      if(xco.ge.xx(1).and.xco.le.xx(nprof).and.ifberg.eq.1) goto 109
c*
  107 call schnitt (xcorn,zcorn,nbody,ncorn,xco,zco,iflag,jj,ii,unit,
     1tili)
c*
      if(tili.eq.1.) goto 109
      do 2 lnx=1,nbody
      if(lnx.ne.jj.and.lnx.ne.lfl(lnx).and.iflag(lnx).eq.1)
     1write(6,60) ii,jj,lnx
  60  format(' Point',i2,' (body',i2,') has reached the border of body'
     1,i2,' !')
      if(lnx.ne.jj.and.lnx.ne.lfl(lnx).and.iflag(lnx).eq.1) goto 109
   2  continue
      goto 111
  109 iuma=1
  111 return
      end
c
c*********************************************************************
      subroutine relbesser(lrun,amin,res,nprof,ream,iumq)
c
c          This subroutine checks if the improvement of E  (E= (O -C ) )
c          is at least 1% or greater than 50% than the previous iteration.
c
      dimension res(1000)
c
      if(lrun.gt.2.and.(amin-res(nprof)).lt.(amin/100).or.
     1ream.gt.(amin-res(nprof))*2) then
                                    iumq=1
      else
                                    ream=amin-res(nprof)
                                    endif
      return
      end
c
```

```
c*********************************************************************
      subroutine waginig
c
c          This subroutine calls the  subroutines necessary to
c          perform the automatic iterative model calculation for
c          gravity. The subroutine calculates and compares the
c          sum of the squares of the residuals and changes the
c          corner-points of the bodies according to the improvement
c          of the model.
c
      common /block1/ nprof,dprof,regfirst,regslope,iobs,h(1000),
     &obs(1000),x(1000),z(1000),robs(1000)
      common /block2/ nbody,xcorn(50,50),zcorn(50,50),ncorn(50),
     &m(50),mi(50),md(50),mx(50),mz(50),rho(50)
      common /block3/ azim,fi,fd,itype,xconv,zconv
      common /block7/ xmov,lfl(50),kfk(50),lrun,amin
      common /block8/ grenze(50),bod(50),jainmax(50),jainmin(50),unitmax
     1(50),grmax(50),grmin(50),itermax(50),unitmin(50),itermin(50),relgr
     2e(50)
      common /block9/ absgre(50),obergr(50),untergr(50),imaxi(50),imini(
     150),imaxex(50),iminex(50),xmaxmax(50),xmaxmin(50),xminmax(50),xmin
     2min(50)
      real m,mi,md,mx,mz
      character*1 ans,absgre*1,relgre*1,cross*1,bod*1,grenze*1
      character*1 imaxex,iminex*1,jainmax*1,jainmin*1
      dimension xc(50),zc(50),xt(1000),zt(1000),resa(1000),res(1000),
     1unit(50,50),xx(1000),zz(1000),iter(50,50),cal(1000)
c
c         Body-questions
c
c*
      call pili(nbody,xcorn,zcorn,ncorn,cross,unit,iter,
     1xx,zz,numtopo,howalti)
c*
      if(cross.eq.'y'.and.numtopo.eq.0) then
                                        do 4 lp=1,nprof
                                        xx(lp)=x(lp)
  4                                     zz(lp)=z(lp)+howalti
                                        endif
      if(numtopo.eq.0) numtopo=nprof
c
c         Body variation loop
c
      do 6 jj=1,nbody
      if(bod(jj).eq.'n') goto 6
      xmov=0.
c
c         Corner variation loop
c
101   do 6 ii=0,ncorn(jj)
      if(xmov.eq.0) goto 103
      if(xcorn(jj,ii).eq.xcorn(jj,imaxi(jj)).and.jainmax(jj).ne.'n'.and.
     1ii.eq.imaxi(jj).or.xcorn(jj,ii).eq.xcorn(jj,imini(jj)).and.
     2jainmin(jj).ne.'n'.and.ii.eq.imini(jj)) goto 103
      goto 6
103   ream=0.
      lrun=0
      if(ii.eq.0) go to 107
      if(iter(jj,ii).eq.0) goto 115
105   lrun=lrun+1
      if(xmov.eq.0.) zcorn(jj,ii)=zcorn(jj,ii)+unit(jj,ii)
      if(ii.ne.imaxi(jj).or.ii.ne.imini(jj)) goto 949
      n113=0
c*
```

```
      call apunkt(imaxi,imini,ii,jj,ncorn,zcorn,unit,n113)
c*
      if(n113.eq.1) goto 113
c
c
  949 if(xmov.eq.1.) xcorn(jj,ii)=xcorn(jj,ii)+unit(jj,ii)
      ium=0
c*
      call agrenze(xmov,jj,ii,xcorn,zcorn,ium)
c*
      if(ium.eq.1) goto 113
c
c       cross-control
c
      if(cross.ne.'y') goto 107
      iuma=0
c*
      call kontrolle(res,xcorn,zcorn,jj,ii,grmin,grmax,nbody,g
     1renze,relgre,unit,ncorn,iuma,xx,zz,numtopo)
c*
      if(iuma.eq.1) goto 113
107   do 3 i=1,nprof
      if(ii.eq.0) resa(i)=0.
      res(i)=0.
      sumi=0.0
c
      do 2 j=1,nbody
      if (rho(j).eq.0.0) go to 2
      nvert=ncorn(j)
      sumj=0.0
c
      do 1 k=1,nvert
      k2=k+1
      if (k.ge.nvert) k2=1
      a1=(xcorn(j,k2)-xcorn(j,k))*xconv
      a2=(zcorn(j,k2)-zcorn(j,k))*zconv
      if (a2.eq.0) a2=.00001
      a=a1/a2
c
      xk=(xcorn(j,k)-x(i))*xconv
      zk=(zcorn(j,k)-z(i))*zconv
      if (zk.eq.0) zk=.00001
c
      xkp1=(xcorn(j,k2)-x(i))*xconv
      zkp1=(zcorn(j,k2)-z(i))*zconv
      if (zkp1.eq.0) zkp1=.00001
c
      b1=xk*zkp1-xkp1*zk
      b=b1/a2
c
      c=b/(1+a**2)
      tk=atan2(xk,zk)
      tkp1=atan2(xkp1,zkp1)
c
      d1=xkp1**2+zkp1**2
      d2=xk**2+zk**2
c
      sumj=sumj+c*(.5*alog(d1/d2)+a*(tkp1-tk))
    1 continue
      sumi=sumi+13.34*rho(j)*sumj
    2 continue
      h(i)=sumi
c
c       calculation of the sum of the squares of the residuals
```

```
c
      if(ii.eq.Ø) then
                  cal(i)=h(i)
                  resa(i)=(robs(i)-cal(i))**2+resa(i-1)
      else
                  res(i)=(robs(i)-h(i))**2+res(i-1)
                  endif
  3   continue
c
c       comparison of the sum of the squares of the residuals
c
      if(ii.eq.Ø) amin=resa(nprof)
      if(ii.eq.Ø) goto 6
      if(res(nprof).ge.amin) goto 113
      iumq=Ø
c*
      call relbesser(lrun,amin,res,nprof,ream,iumq)
c*
      amin=res(nprof)
      amina=(amin/nprof)**Ø.5
      if(lrun.eq.1.and.ii.eq.1) print'(/)'
      if(iumq.eq.1) write(6,4Ø)amina,lrun,iter(jj,ii),jj,ii,xcorn(jj,ii)
     1,zcorn(jj,ii)
4Ø    format(' deviation:',f8.2,'  iteration:',i2,'/',i1,'  body/point
     1:',i2,'/',i2,'   x:',f8.1,'   z:',f8.1,'*')
      if(iumq.eq.Ø) write(6,6Ø)amina,lrun,iter(jj,ii),jj,ii,xcorn(jj,ii)
     1,zcorn(jj,ii)
6Ø    format(' deviation:',f8.2,'  iteration:',i2,'/',i1,'  body/point
     1:',i2,'/',i2,'   x:',f8.1,'   z:',f8.1)
      if(iumq.eq.1.or.lrun.ge.iter(jj,ii)) goto 115
      goto 1Ø5
 113  if(xmov.eq.Ø) then
                  zcorn(jj,ii)=zcorn(jj,ii)-unit(jj,ii)
      else
                  xcorn(jj,ii)=xcorn(jj,ii)-unit(jj,ii)
                  endif
      do 8 lr=1,nbody
      if(xmov.eq.Ø) then
         zcorn(lfl(lr),kfk(lr))=zcorn(lfl(lr),kfk(lr))-unit(jj,ii)
      else
        xcorn(lfl(lr),kfk(lr))=xcorn(lfl(lr),kfk(lr))-unit(jj,ii)
      endif
8     continue
      if(lrun.eq.1) then
                  unit(jj,ii)=-unit(jj,ii)
                  iter(jj,ii)=iter(jj,ii)+1
                  goto 1Ø5
                  endif
115   if(ii.ne.ncorn(jj).or.xmov.eq.1) goto 6
      unit(jj,imaxi(jj))=unitmax(jj)
      unit(jj,imini(jj))=unitmin(jj)
      iter(jj,imaxi(jj))=itermax(jj)
      iter(jj,imini(jj))=itermin(jj)
      xmov=1.
      goto 1Ø1
 6    continue
      print'(//,a,/)',' Results ...'
      write(6,8Ø) ((j,i,xcorn(j,i),zcorn(j,i),i=1,ncorn(j)),j=1,nbody)
  8Ø  format(' body ',i2,'  point ',i2,'   x:',f9.2,'   z:',f9.2)
      print'(/)'
      return
      end
c
c*************************************************************************
```

```
      subroutine apunkt(imaxi,imini,ii,jj,ncorn,zcorn,unit,n113)
c
c        This subroutine prevents the z-coordinate of the outermost
c        points from exceeding the z-coodinates of the adjacent corner-points,
c        if the z-coordinate of the outermost point had been between
c        the two values in the starting model.
c
      dimension imaxi(50),imini(50),ncorn(50),zcorn(50,50),unit(50,50)
      if(ii.ne.imaxi(jj)) goto 373
      iop=imaxi(jj)+1
      ion=imaxi(jj)-1
      if(iop.gt.ncorn(jj)) iop=1
      if(ion.lt.1) ion=ncorn(jj)
      if(zcorn(jj,imaxi(jj)).le.zcorn(jj,ion).and.(zcorn(jj,
     1imaxi(jj))-unit(jj,imaxi(jj))).gt.zcorn(jj,ion).or.zcorn
     2(jj,imaxi(jj)).ge.zcorn(jj,iop).and.(zcorn(jj,imaxi(jj))
     3-unit(jj,imaxi(jj))).lt.zcorn(jj,iop)) goto 113
373   if(ii.ne.imini(jj)) goto 114
      ioop=imini(jj)+1
      ioon=imini(jj)-1
      if(ioop.gt.ncorn(jj)) ioop=1
      if(ioon.lt.1) ioon=ncorn(jj)
      if(zcorn(jj,imini(jj)).ge.zcorn(jj,ioon).and.(zcorn(jj,
     1imini(jj))-unit(jj,imini(jj))).lt.zcorn(jj,ioon).or.zcorn
     2(jj,imini(jj)).le.zcorn(jj,ioop).and.(zcorn(jj,imini(jj))
     3-unit(jj,imini(jj))).gt.zcorn(jj,ioop)) goto 113
      goto 114
113   n113=1
114   return
      end
c
c
c*********************************************************************
      subroutine schnitt(xcorn,zcorn,nbody,ncorn,x,y,iflag,jj,ii,unit,
     1tili)
c
c        This subroutine compares the corner-points of the polygon
c        to determine if the sides are twisted or if a corner-point of
c        one body had been placed inside another body.
c
      common /block7/ xmov,lfl(50),kfk(50),lrun,amin
      dimension xcorn(50,50),ncorn(50),iflag(50),zcorn(50,50),
     1unit(50,50),xpoly(50),ypoly(50)
c
      tili=0.
      do 5 l=1,nbody
      kfk(l)=0
      lfl(l)=0
      if(xmov.eq.0..and.l.eq.jj) zcorn(jj,ii)=zcorn(jj,ii)-unit(jj,ii)
      if(xmov.eq.1..and.l.eq.jj) xcorn(jj,ii)=xcorn(jj,ii)-unit(jj,ii)
      npoly=ncorn(l)
      do 4 k=1,npoly
      if(xmov.eq.0..and.xcorn(l,k).eq.x.and.zcorn(l,k).eq.(y-unit(jj,ii)
     1).and.l.ne.jj) goto 6
      if(xmov.eq.1..and.xcorn(l,k).eq.(x-unit(jj,ii)).and.zcorn(l,k).eq.
     1y.and.l.ne.jj) goto 6
      goto 7
6     zcorn(l,k)=y
      xcorn(l,k)=x
      kfk(l)=k
      lfl(l)=1
7     xpoly(k)=xcorn(l,k)
      ypoly(k)=zcorn(l,k)
4     continue
```

```
      icount=0
c
c
c
      xmax=amax1(xpoly(1),xpoly(npoly))
      xmin=amin1(xpoly(1),xpoly(npoly))
      if(1.ne.jj.or.xmov.eq.1) goto 113
      ymax=amax1(ypoly(1),ypoly(npoly))
      ymin=amin1(ypoly(1),ypoly(npoly))
      if(x.eq.xmin.and.x.eq.xmax.and.unit(jj,ii).gt.0.and.y.gt.ymax.
     1and.(y-unit(jj,ii)).lt.ymax) tili=1.
      if(x.eq.xmin.and.x.eq.xmax.and.unit(jj,ii).lt.0.and.y.lt.ymin.
     1and.(y-unit(jj,ii)).gt.ymin) tili=1.
113   if(x.le.xmin.or.x.gt.xmax) goto 2
      xdel=xpoly(1)-xpoly(npoly)
      if(xdel.lt.0.0001.and.xdel.gt.(-0.0001)) xdel=0.0001
      a=(ypoly(1)-ypoly(npoly))/xdel
      b=ypoly(1)-a*xpoly(1)
      y0=a*x+b
      if(1.ne.jj.or.xmov.eq.1) goto 111
      if(xpoly(npoly).eq.xcorn(jj,ii).and.ypoly(npoly).eq.zcorn(jj,ii))
     1goto 111
      if(unit(jj,ii).gt.0.and.y.gt.y0.and.(y-unit(jj,ii)).lt.y0) tili=1.
      if(unit(jj,ii).lt.0.and.y.lt.y0.and.(y-unit(jj,ii)).gt.y0) tili=1.
111   if(y0.le.y) goto 2
      icount=icount+1
 2    continue
c
c
c
      do 1 i=2,npoly
      xmax=amax1(xpoly(i),xpoly(i-1))
      xmin=amin1(xpoly(i),xpoly(i-1))
      if(1.ne.jj.or.xmov.eq.1) goto 115
      ymax=amax1(ypoly(i),ypoly(i-1))
      ymin=amin1(ypoly(i),ypoly(i-1))
      if(x.eq.xmin.and.x.eq.xmax.and.unit(jj,ii).gt.0.and.y.gt.ymax.
     1and.(y-unit(jj,ii)).lt.ymax) tili=1.
      if(x.eq.xmin.and.x.eq.xmax.and.unit(jj,ii).lt.0.and.y.lt.ymin.
     1and.(y-unit(jj,ii)).gt.ymin) tili=1.
115   if(x.le.xmin.or.x.gt.xmax) goto 1
      a=(ypoly(i)-ypoly(i-1))/(xpoly(i)-xpoly(i-1))
      b=ypoly(i)-a*xpoly(i)
      y0=a*x+b
      if(1.ne.jj.or.xmov.eq.1) goto 117
      if(xpoly(i-1).eq.xcorn(jj,ii).and.ypoly(i-1).eq.zcorn(jj,ii))
     1goto 117
      if(unit(jj,ii).gt.0.and.y.gt.y0.and.(y-unit(jj,ii)).lt.y0) tili=1.
      if(unit(jj,ii).lt.0.and.y.lt.y0.and.(y-unit(jj,ii)).gt.y0) tili=1.
117   if(y0.le.y) goto 1
      icount=icount+1
 1    continue
c
c
c
      iflag(1)=icount-2*(icount/2)
      if(xmov.eq.0..and.1.eq.jj) zcorn(jj,ii)=zcorn(jj,ii)+unit(jj,ii)
      if(xmov.eq.1..and.1.eq.jj) xcorn(jj,ii)=xcorn(jj,ii)+unit(jj,ii)
  5   continue
      return
      end
c
c*********************************************************************
```

```
      subroutine berg(xpoly,ypoly,npoly,x,y,ifberg)
c
c        This subroutine compares the corner-points of the polygon
c        with the observation-points to determine if
c
c        1) a body extends above the topographic surface,
c
c        2) a corner-point of the polygon exceeds the upper or lower
c           vertical limit relative to the topography.
c
      dimension xpoly(npoly),ypoly(npoly)
      icount=0
c
c
c
      xmax=amax1(xpoly(1),xpoly(npoly))
      xmin=amin1(xpoly(1),xpoly(npoly))
      if(x.le.xmin.or.x.gt.xmax) goto 2
      del=xpoly(1)-xpoly(npoly)
      if(xdel.lt.0.0001.and.xdel.gt.(-0.0001)) xdel=0.0001
      a=(ypoly(1)-ypoly(npoly))/del
      b=ypoly(1)-a*xpoly(1)
      y0=a*x+b
      if(y0.le.y) goto 2
      icount=icount+1
 2    continue
c
c
c
      do 1 i=2,npoly
      xmax=amax1(xpoly(i),xpoly(i-1))
      xmin=amin1(xpoly(i),xpoly(i-1))
      if(x.le.xmin.or.x.gt.xmax) goto 1
      a=(ypoly(i)-ypoly(i-1))/(xpoly(i)-xpoly(i-1))
      b=ypoly(i)-a*xpoly(i)
      y0=a*x+b
      if(y0.le.y) goto 1
      icount=icount+1
 1    continue
c
c
      ifberg=icount-2*(icount/2)
      return
      end
c
c*************************************************************************
      subroutine write_topo(xx,zz,numtopo)
c
c        This subroutine writes topographic data to an ascii-file.
c
      dimension xx(1000),zz(1000)
      character*40 filename,quest*80
c
      print'(a)',' '
      quest='Name of topo-file to write to'
      idumb=iaquest(quest,filename,'(a40)',40)
      open(34,file=filename,form='formatted',status='unknown',
     1carriagecontrol='list')
      write(34,*) numtopo
      write(34,*) (xx(j),zz(j),j=1,numtopo)
      close (34)
      return
      end
c
```

```
c*************************************************************************
      subroutine read_topo(xx,zz,numtopo)
c
c         This subroutine reads topographic data from an ascii-file.
c
      dimension xx(1000),zz(1000)
      character*40 filename,quest*80
c
      print'(a)',' '
      quest='Name of topo-file'
      idumb=iaquest(quest,filename,'(a40)',40)
      open(35,file=filename,form='formatted',status='old',
     1carriagecontrol='list')
      read (35,*) numtopo
      read (35,*) (xx(j),zz(j),j=1,numtopo)
      close (35)
      return
      end
```