THE UNITED NATIONS UNIVERSITY

ORKUSTOFNUN
NATIONAL ENERGY AUTHORITY

ONE DIMENSIONAL INVERSION OF SCHLUMBERGER RESISTIVITY SOUNDINGS

Computer Program, Description and User's Guide

Knútur Árnason
Gylfi Páll Hersir

Geothermal Training Programme
Reykjavík, Iceland
Report 8, 1988

ABSTRACT

This report describes a computer program for one-dimensional inversion of Schlumberger resistivity soundings. It is also meant as a user's guide to the program. A short description of the inversion method is followed by step by step instructions of how to run the program. The input and output files are described as well as plotting utilities for graphic display of the results. Finally all the source files of the program are listed in an appendix. Two 5-1/4" diskettes containing all files (source files and run files) are available upon request from the Geothermal Training Programme.

CONTENTS

1. INTRODUCTION

The following is a user's guide and a short description of the program SLINV (SchLumberger INVersion) for one-dimensional inversion of Schlumberger resistivity soundings. SLINV is a non-linear least-squares inversion program using a Levenberg-Marquardt inversion algorithm together with a fast forward routine based on the linear filter method. The Levenberg-Marquardt inversion algorithm used in the program is described by H. K. Johansen (1977) and the forward algorithm for calculating the response of a given one-dimensional model is described by H. K. Johansen (1975).

The program consists of ten source files, the main program SLINV.FOR and nine subroutines. The modules are listed in Table 1 and the source code is given in appendix. The program is written in standard FORTRAN 77.

MODULE	FUNCTION
SLINV	Main program; inversion algorithm
RDAT	Reads input data from file and terminal
FLT	Stores J1 Hankel transform filter
SLFW	Forward algorithm; calculates apparent resistivity curve from a given model
CHSQ	Calculates chi-square sum
SLDR	Calculates partial derivative matrix
SVDC	Performs a singular value decomposition on the partial derivative matrix
ORDW	Orders eigenvalues in increasing order
NEWP	Calculates increments to be added to model parameters to get a new model
WROT	Writes out results into output files

Table 1. The modules of the program SLINV

The executable file SLINV.EXE was made by compiling the source files by the Microsoft FORTRAN Optimizing Compiler, version 4.01, and linking by the Microsoft Object Linker. The program can be run on IBM PC or PC-compatible computers both with or without an 8087 coprocessor but it runs about 24 times faster with an 8087 coprocessor. The computing time for inversion of 20 data points (two decades) in terms of a 3-layered model is about 4.3 sec. per iteration step with an 8087 coprocessor but about 105 sec. without an 8087 coprocessor. The program can be made to run on main frame computers by compiling and linking the FORTRAN source files but the plotting utilities (described below) only run on IBM PC and PC-compatibles.

The inversion program reads measured apparent resistivity data from an input file and prompts for an initial guess for a one-dimensional resistivity model. Then it iteratively adjusts the resistivity model to minimize the difference between the measured and the calculated apparent resistivity values. The results are written into two files, an output list file that can be typed on the screen and printed as a hard copy and an output plot file that can be plotted both on the screen and as a hard copy on a printer or a plotter. In order to run the inversion program, the file SLINV.EXE must be either in the working directory or a directory specified in the PATH-list. To plot the results the files SPLOT.BAT (screen plot), PPLOT.BAT (paper plot), RES.GRD (grid file) and CENTERED.SYM (plot symbols) must be in the working directory, and the files VIEW.EXE (screen plot) and PLOT.EXE (paper plot) must either be in the working directory or a directory specified in the PATH-list.

A slightly modified version of the forward routine of the inversion program has been made into a separate program called SLUM (the source files are SLUM.FOR for the main program and FLT.FOR and SLWROT.FOR for the subroutines; the executable file is SLUM.EXE). SLUM calculates the apparent resistivity curve for a one-dimensional resistivity model which is read from the terminal and for AB/2 (half the current electrode spacing) values equally distributed on log-scale, with ten points per decade, over an interval specified by the user. The results are written into an output file (that has the right format for an input file for SLINV) and a plot file that can be plotted in the same way as the output plot files from the inversion program SLINV. The program SLUM is a by-product of the inversion program, meant for simple model calculations and will not be discussed further.

2. THE STRUCTURE OF THE PROGRAM

The backbone of the program SLINV is a general non-linear least-squares inversion algorithm of the Levenberg-Marquardt type. The inversion algorithm is supplemented by routines for data input, RDAT, and output, WROT, a forward routine, SLFW, that calculates the apparent resistivity values from a given one-dimensional resistivity model and a routine, SLDR, that calculates the partial derivatives of the apparent resistivity with respect to the model parameters. The general structure of the program is shown on Figure 2.1. The least-squares algorithm, the forward routine SLFW, and the partial derivative routine SLDR will now be discussed briefly.

2.1 The inversion algorithm

The program SLINV, like most inversion programs, works in such a way that it reads the measured data points (apparent resistivity curve) and prompts for a starting model. The interpreter guesses, by visual inspection of the data curve, the number of layers and initial model parameters i.e. the resistivity values and thicknesses of the layers. Each model parameter can either taken to be a free or a fixed parameter. The program iteratively adjusts the values of the free model parameters to get the best fit between the measured curve and the curve calculated from the model. It is important to realize that the program does not change the number of layers during the iteration process. It is therefore in most cases necessary to try models with different numbers of layers to find the model that best fits the data. It should also be kept in mind that the model resulting from the iterative inversion can depend on the initial guess. A poor guess can lead the inversion process astray.

In the inversion algorithm, all computations are done with data and model parameters on logarithmic form, that is to say ($\ln(AB/2),\ln(\rho_a)$) is used instead of ($AB/2,\rho_a$), where AB/2 is half the current electrode spacing. The model parameters are kept as $P(i)=\ln(p(i))$, where the p(i) stand for the resistivity values and layer thicknesses. This is done because the non-linearity in the dependence of the apparent resistivity on the model parameters is not as severe in the logarithmic as in the linear representation. The logarithmic representation furthermore prevents the occurrence of non-physical negative model parameters.

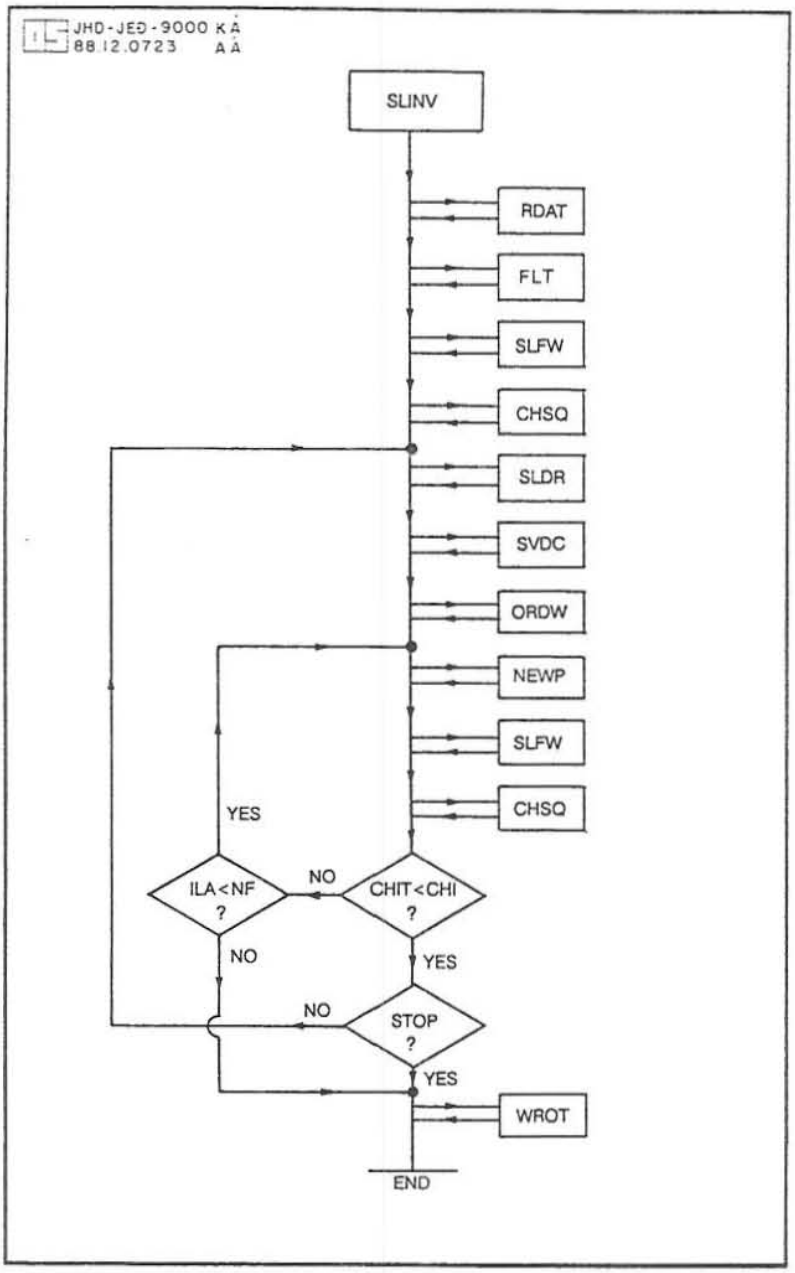

Figure 2.1 Structure of inversion program

The quality of the fit between the measured and the calculated apparent resistivity values, calculated by the subroutine SLFW, is measured by the chi-square sum, CHI, which is calculated by the subroutine CHSQ. CHI is given in terms of the natural logarithms of the measured and calculated apparent resistivity values by the following formula:

$$CHI = \left[\frac{1}{ND}\sum_{1}^{ND}[(\ln(\rho_{am}) - \ln(\rho_{ac}))\cdot WPM]^2\right]^{1/2},$$

where ND is the number of data points and WPM is a weight factor (to be discussed later). The lower the CHI is, the better is the fit. If CHI is less than 0.1 it can be interpreted as the average fractional difference between the measured and the calculated apparent resistivity values.

The program always keeps the best model obtained. Each iteration cycle starts with the determination of a temporary model to be tried next. To determine the temporary model, the partial derivatives of the apparent resistivity values, with respect to the free model parameters, are calculated by the subroutine SLDR. The partial derivative matrix, **A**, is decomposed, by the singular value decomposition routine SVDC, into a product of an orthogonal data eigenvector matrix, **U**, a diagonal eigenvalue matrix, **L**, and the transpose of an orthogonal parameter eigenvector matrix, **V**:

$$\mathbf{A} = \mathbf{U}\cdot\mathbf{L}\cdot\mathbf{V}^t.$$

Let the logarithms of the free model parameters and the calculated apparent resistivity values be represented by the vectors **p** and **q** respectively. A small variation **dp** of the model parameters results in a variation **dq** of the calculated apparent resistivity values given as:

$$\mathbf{dq} = \mathbf{A}\cdot\mathbf{L}\cdot\mathbf{V}^t\cdot\mathbf{dp}.$$

To get the increments to be added to the model parameter vector **p** in order to get the temporary model to be tried next, the vector **dq** in the equation above is taken to be the difference between the measured and calculated apparent resistivity values. The equation is "inverted" in the subroutine NEWP by multiplying **dq** by a "damped" inverse of the partial derivative matrix. The damping is performed by adding a Marquardt parameter to the eigenvalues. This damping is necessary because if the partial derivative matrix is nearly singular, one or more of the eigenvalues are very small, and an undamped inversion of the equation would result in unreasonably large increments **dp**. The Marquardt parameter XLA is taken to be equal to one of the eigenvalues which have been ordered in an increasing order in the subroutine ORDW. The ordered eigenvalues are numbered by the index ILA which runs from 1 to NF, the number of free parameters in the model.

The first temporary model tried in each iteration step is obtained by adding the increments resulting from the damped inversion of the above equation, using the smallest eigenvalue (but not smaller than 0.01, the index ILA shows which eigenvalue is used) as a Marquardt parameter. If the chi-square sum, CHIT, for the temporary model is

less than CHI then the temporary model is kept as the best model and a new iteration cycle is started. If CHIT is higher than CHI, another temporary model, obtained by inversion of the above equation with increased damping with a higher eigenvalue (ILA increased by one), is tried. This is continued until CHIT becomes lower than CHI or ILA gets higher than NF, in which case the iteration process is terminated.

There are four stop checks in the program. The iteration process terminates if:

a) The parameter ILA gets higher than NF. This is called no-convergence. If the program terminates on this stop check it does not necessarily mean that an acceptable fit to the measured data was not obtained. It simply states that the program could not further improve the fit.
b) The average fractional difference, CHI, between the measured and the calculated apparent resistivity values becomes less than 10^{-3} i.e. an average difference less than 0.1%. This is called CHI-convergence.
c) The fractional decrease in CHI in the last iteration is less than 10^{-5}. This is called DCHI-convergence.
d) The program has performed the number of iterations that the operator asked for and he does not wish to perform more iterations. This stop check is called maximum of iterations.

A CHI-convergence is seldom obtained in inversion of real measured data and the iteration is usually terminated on no-convergence or DCHI-convergence (no further improvement) or on maximum of iterations.

The forward routine SLFW calculates the apparent resistivity values at AB/2 values equally distributed on logarithmic scale with ten points per decade. The inversion algorithm demands therefore that the data have this distribution. If the measured data do not fulfill this condition, an interpolation, on log-log scale, is performed (in the data input routine RDAT) in order to get data points, equally spaced in logarithm of AB/2, with ten points per decade.

A different weight can be given to the data points in the inversion. The data points can be weighed either with respect to the apparent resistivity value (WMOD=RE) or with respect to the current electrode spacing (WMOD=AB). In both cases the degree of the weighing can further be controlled by the weight parameter RW ($-1 \leq RW \leq 1$). Using RW=1, the data points are weighed with WPM proportional to $\ln(\rho_a)$ or

ln(AB/2) for WMOD equal to RE or AB, respectively. This causes data points with higher ρ_a, in the case WMOD=RE, or higher AB/2, in the case WMOD=AB, to have more weight in the inversion. For WR=-1 the data points are weighed with WPM inversely proportional to $\ln(\rho_a)$ or ln(AB/2) depending on WMOD, giving the lower values more weight. For WR in between -1 and 1 the data is weighed with WPM proportional to $(\ln(\rho_a))^{WR}$ or $(\ln(AB/2))^{WR}$ giving weights in between the above extreme cases. For WR=0 all data points are treated on equal footing for both cases of weighing mode WMOD. The different modes of weighing can be used to put emphasis on different parts of the measured data curve. If we want to emphasize the part of the data curve with high AB/2 values, we choose WMOD=AB and WR close to 1. Similarly if we want to emphasize the low apparent resistivity values in the curve, we take WMOD=RE and WR close to -1. Normally it is recommended to give all data points similar weight and hence to take WR close to 0.

2.2 The forward algorithm

The forward algorithm SLFW calculates the apparent resistivity values for a given one-dimensional resistivity model. It uses the gradient approximation for calculating the apparent resistivity. This means that it is assumed that the receiving dipole is infinitesimally short compared to the transmitter dipole. This implies that the actual electrode configuration in the sounding is not simulated and only one apparent resistivity value can be assigned to each value of AB/2.

The apparent resistivity, as a function of half the current electrode spacing, AB/2 = r, is given by the following formula:

$$\rho_a(r) = r^2 \int_0^\infty K(\lambda)\, J_1(\lambda r)\, \lambda\, d\lambda .$$

The kernel function $K(\lambda)$ contains the model parameters. The Hankel transform integral is calculated by the use of a digital filter. The filter is transferred through a common block from the subroutine FLT which is called by the main program

2.3 The partial derivative algorithm

The partial derivatives of the logarithms of the apparent resistivity values with respect to the logarithms of the model parameters are calculated by the subroutine SLDR.

The elements of the partial derivative matrix are given by the formula:

$$A_{i,j} = \frac{r_i^2}{\rho_a(r_i)} \int_0^\infty \frac{\partial K(\lambda)}{\partial \ln(p_j)} J_1(\lambda r_i)\, \lambda\, d\lambda .$$

The Hankel transform is calculated by using the digital filter stored in FLT.

3. RUNNING THE PROGRAM

In order to run the inversion program, the file SLINV.EXE must either be in the working directory or a directory specified in the PATH-list. The measured apparent resistivity curve is read from an input file. The input file must have one data point in each line. A data point consists of a pair of numbers separated by a comma (,), a space(s) or a tab. The first number is half the current electrode spacing, AB/2, and the second number is the corresponding measured apparent resistivity value, ρ_a.

The following is an example of an input data file:

```
  1.26      74.44
  1.58      60.51
  2.00      44.00
  2.51      27.79
  3.16      14.93
  3.98       7.17
  5.01       3.92
  6.31       3.33
  7.94       3.85
 10.00       4.76
 12.59       5.92
 15.85       7.35
 19.95       9.09
 25.12      11.21
 31.62      13.76
 39.81      16.80
 50.12      20.39
 63.10      24.56
 79.43      29.34
100.00      34.72
```

When a data file containing the measured apparent resistivity curve has been created, having the format described above, the inversion program is run by going through the following steps. (All user's responses discussed below are to be followed by striking the return key):

1. The inversion program is started by typing SLINV at the system prompt.

2. The program prompts for input file name. This is answered by typing the name of the file containing the measured data curve to be inverted.

3. The program now asks how the data should be weighed. If the data is to be weighed with respect to the apparent resistivity values then type RE but type AB if it is to be weighed with respect to AB/2. The default is RE.

4. The program asks for the weight parameter RW discussed above. This is answered by typing a number in the interval -1 to 1. It is generally recommended to use RW close to 0.

5. The program asks for the guessed initial model. It asks for the number of layers which is answered by typing the guessed number of layers. Next it asks for the model parameters for each layer (resistivity in Ohmm and thickness in m). Each parameter can either be a free parameter to be adjusted by the program or a fixed parameter not to be adjusted. For a free parameter the guessed value is typed. For parameters to be held fixed the corresponding parameter values are followed by a comma and an asterisk (e.g. 235,*).

6. The program prompts for the number of iterations to be performed. This is answered by typing the desired number of iterations.

7. Finally the program prompts for names of an output list file and an output plot file. The results from the iterative inversion are written into these files and will be discussed later. When these file names have been specified the program starts the iteration process.

During the iterations the program writes on the terminal the iteration number ITR, the best model parameters obtained so far (resistivity values, rho and layer thicknesses, d) and the temporary model parameters (rhot and dt) to be tried next. It also writes the lowest CHI-value obtained and the temporary CHIT obtained from the temporary model. If CHIT is lower than CHI then the temporary model is kept as the best model and the program proceeds to the next iteration with a new temporary model. If CHIT is higher than CHI another temporary model is tried. This is repeated until CHIT becomes lower than CHI or ILA, which is displayed on the screen, becomes higher than NF, the number of free parameters in the model in which case the program terminates on the criterion of no-convergence.

If the iteration process has not stopped on the no-convergence, the CHI-convergence or the DCHI-convergence stop checks (see above) and the number of iterations specified (in step 6 above) has been performed, the program pauses and asks if the iteration process is to be continued. If this question is answered with N (no), the iteration process is terminated on maximum number of iterations. If it is answered with Y (yes), the program asks how many more iterations are to be performed.

4. OUTPUT FILES

The results from the inversion are written into two output files, the output list file and the output plot file. The content of these files will be discussed shortly.

4.1 The output list file

The following is an example of an output list file:

```
WEIGHING MODE = RE    WEIGTH PARAMETER RW =  .0

INITIAL MODEL PARAMETERS:

rho: 150.000      .500  120.000
  d:   2.000    3.000
 CHI= .12476E+01

ITR= 1  ILA= 2  CHI= .5430E+00  DCHI= .5648E+00
rho:   39.69     1.56    18.02
  d:    1.85     2.79

ITR= 2  ILA= 2  CHI= .2343E+00  DCHI= .5684E+00
rho:   61.74     1.69    35.30
  d:    1.03     2.56

ITR= 3  ILA= 1  CHI= .7109E-01  DCHI= .6967E+00
rho:   99.79     1.41    68.72
  d:    1.00     2.92

ITR= 4  ILA= 1  CHI= .5725E-02  DCHI= .9195E+00
rho:  100.70     1.28    96.27
  d:     .99     2.57

ITR= 5  ILA= 1  CHI= .1865E-02  DCHI= .6742E+00
rho:  100.44     1.18   100.25
  d:     .99     2.37

ITR= 6  ILA= 1  CHI= .1443E-02  DCHI= .2261E+00
rho:  100.22     1.10   100.11
  d:    1.00     2.21

ITR= 7  ILA= 1  CHI= .1302E-02  DCHI= .9785E-01
rho:  100.06     1.04    99.93
  d:    1.00     2.08
```

```
ITR= 8  ILA= 2  CHI= .1296E-02  DCHI= .4711E-02
rho:  100.08      1.04     99.95
  d:    1.00      2.08

ITR= 9  ILA= 2  CHI= .1293E-02  DCHI= .2228E-02
rho:  100.07      1.04     99.94
  d:    1.00      2.07

*** THE PROGRAM TERMINATED  AFTER 10 ITRERATIONS ***

ISTOP=2 * MAX ITERATIONS *

FINAL CHI-SQ. SUM IS CHI= .1290E-02
```

THE FINAL MODEL PARAMETERS ARE:

```
rho:  100.06      1.03     99.93
  d:    1.00      2.06
```

DATA EIGENVECTORS:

1	-.148	.051	-.614	.059	.367
2	-.207	.068	-.499	.034	.136
3	-.287	.089	-.340	.001	-.124
4	-.386	.113	-.133	-.038	-.338
5	-.489	.126	.113	-.070	-.356
6	-.531	.086	.330	-.052	.033
7	-.377	-.074	.312	.089	.576
8	-.140	-.242	.085	.242	.374
9	-.057	-.292	-.019	.272	-.014
10	-.045	-.296	-.035	.249	-.133
11	-.044	-.293	-.036	.214	-.146
12	-.043	-.290	-.036	.172	-.140
13	-.043	-.285	-.036	.121	-.128
14	-.042	-.280	-.036	.060	-.111
15	-.041	-.274	-.036	-.012	-.088
16	-.040	-.267	-.036	-.096	-.060
17	-.038	-.259	-.036	-.196	-.026
18	-.037	-.249	-.036	-.311	.015
19	-.035	-.237	-.036	-.442	.063
20	-.033	-.225	-.036	-.591	.117
	1	2	3	4	5

PARAMETER EIGENVECTORS:

1	-.302	.095	-.948	.034	.017
2	-.160	-.687	-.002	.080	.704
3	-.011	-.092	-.041	-.995	.021
4	-.929	.194	.314	-.021	-.019
5	.142	.687	.035	-.051	.710
	1	2	3	4	5

PARAMETER EIGENVALUES:

6.474 4.431 1.170 .333 .021

CORRELATION MATRIX:

1	1.000				
2	.713	1.000			
3	.150	.311	1.000		
4	-.867	-.945	-.241	1.000	
5	.711	1.000	.322	-.944	1.000
	1	2	3	4	5

I	AB/2	Rhoam	Rhoac	WPM
1	1.26	74.50	74.44	1.00
2	1.58	60.26	60.48	1.00
3	2.00	44.14	43.97	1.00
4	2.51	27.75	27.75	1.00
5	3.16	14.90	14.91	1.00
6	3.98	7.16	7.16	1.00
7	5.01	3.92	3.92	1.00
8	6.31	3.33	3.33	1.00
9	7.94	3.85	3.85	1.00
10	10.00	4.76	4.76	1.00
11	12.59	5.92	5.92	1.00
12	15.85	7.35	7.35	1.00
13	19.95	9.09	9.09	1.00
14	25.12	11.21	11.21	1.00
15	31.62	13.76	13.76	1.00
16	39.81	16.80	16.80	1.00
17	50.12	20.39	20.39	1.00
18	63.10	24.56	24.56	1.00
19	79.43	29.34	29.34	1.00
20	100.00	34.72	34.71	1.00

The program begins by writing the weighing mode and the weight parameter as well as the initial model and the corresponding chi-square sum. During the iteration process the program writes for each iteration the iteration number ITR, the Marquardt parameter counter ILA, the chi-square sum CHI for the model obtained in the present iteration, the fractional decrease DCHI of the chi-square sum between the present and the last iteration and the model obtained in the present iteration step. Model parameters that are held fixed are followed by an asterisk. When the iteration process is finished the program writes the number of iterations performed and the stop check on which the program stopped. Then it writes the final chi-square sum and the best model obtained.

Next the program writes out information on how changes in the model parameters affect the calculated apparent resistivity values. This is described by three matrices **U**, **L** and **V**. **U** is an ND×NF matrix whose column vectors are the data eigenvectors listed (as columns) in the output list file (ND and NF are the number of data points and free model parameters). **L** is an NF×NF diagonal matrix whose diagonal elements are the parameter eigenvalues written in the list file. **V** is an NF×NF orthogonal matrix whose column vectors are the listed parameter eigenvectors (as columns).

In order to bring out the significance of these matrices we think of the resistivity model as an NF dimensional vector **p**. If no parameter is fixed, the first NL components of this vector are the natural logarithms of the resistivity values of the layers (NL is the number of layers in the model, NF=2·NL-1 if no parameter is fixed). The remaining NL-1 components are the natural logarithms of the layer thicknesses. Fixed parameters are not included in the vector **p**. For example, if we take a three layered model and fix the resistivity of the second layer then NF=4 and p_1 and p_2 are the natural logarithms of the resistivity values of the first and the third layer and p_3 and p_4 are the natural logarithms of the thicknesses of the first and the second layer. In the same way we think of the natural logarithms of the calculated apparent resistivity values as an ND dimensional vector **q**. If we change the model vector **p** by the amount **dp** then the calculated apparent resistivity vector will be changed by **dq** according to the following equation:

$$\mathbf{dq} = \mathbf{U} \cdot \mathbf{L} \cdot \mathbf{V}^t \cdot \mathbf{dp}$$

where $\mathbf{V}^t$ means the transpose of the matrix **V**.

From this equation the significance of the data and parameter eigenvectors and the eigenvalues can be deduced. In the first place we see that the apparent resistivity increments dq are proportional to the eigenvalues. The first parameter eigenvector shows which model parameters have the strongest association with the first eigenvalue which is normally the highest one. The degree of association is shown by the absolute value of the components of the eigenvectors, the higher the absolute value the greater the contribution of the corresponding model parameter. Likewise the eigenvector corresponding to the smallest eigenvalue shows which model parameters have the least influence on the calculated apparent resistivity. The model parameters that are most strongly associated to the highest eigenvalue are the most reliable ones whereas the parameters associated to the smallest eigenvalue are the most uncertain parameters in the final model. The relative contribution of the parameter eigenvectors to the different apparent resistivity values is described by the data eigenvectors.

In the three layer example above we see that the thickness of the first layer is the best determined model parameter because the eigenvector corresponding to the highest eigenvalue has the absolute value of the fourth component close to one while the other components are relatively small. We also see that the resistivity value and the thickness of the second layer are not well determined because they have their greatest contribution to the fifth eigenvector (components two and five) which is associated to the smallest eigenvalue. Furthermore we see that the contributions of these parameters to the last eigenvector are similar in magnitude and have the same sign (0.704 and 0.710). This implies that it is only the ratio between the thickness and the resistivity of the second layer that is determined to some degree of reliability.

This layer is an example of what is called an equivalence layer of the s-type. Equivalence layer of the s-type occurs when a relatively thin layer of low resistivity exists between layers of considerably higher resistivities. For such a layer, it is only the longitudinal conductance (the ratio of the thickness and the resistivity) that can be determined with some accuracy. Another type of equivalence layers, called t-type equivalence layers, is also common in one-dimensional resistivity models. They occur when a relatively thin layer with high resistivity is over- and underlain by considerably lower resistivities. For these layers it is only the transverse resistance (the product of the thickness and the resistivity) that is determined with some accuracy. If a t-type equivalence layer is present, its model parameters (resistivity and thickness) have their

greatest contribution to the parameter eigenvector associated to the smallest eigenvalue and the contributions are similar in magnitude but with opposite signs.

Equivalence layers can also be identified by observing the parameter correlation matrix which is written in the output list file. If an off-diagonal element measuring the correlation between a pair of model parameters is close to 1, then only the ratio between the corresponding parameters is fairly well determined (s-type equivalence). If, on the other hand, an off-diagonal element is close to -1, then only the product of the corresponding parameters is determined (t-type equivalence). In the example above we see that the correlation matrix element corresponding to parameters number two and five (resistivity and thickness of the second layer) is equal to 1. This shows that the second layer is an equivalence layer of the s-type.

Finally the program writes into the output list file the AB/2 values for the data points, the measured apparent resistivity values, the apparent resistivity values calculated from the final model and the weight parameters of the data points.

4.2 The output plot file

The inversion program writes the measured data points, the calculated apparent resistivity values, the final model and the value of the chi-square sum into the output plot file. The content of this file can be plotted both on the terminal and as a hard copy on a printer or a plotter. To plot the results, the files SPLOT.BAT (screen plot), PPLOT.BAT (paper plot), RES.GRD (grid file) and CENTERED.SYM (plot symbols) must be in the working directory and the files VIEW.EXE (screen plot) and PLOT.EXE (paper plot) must be either in the working directory or a directory specified in the PATH-list.

The measured apparent resistivity values ar plotted as small circles on a double logarithmic plot and the calculated apparent resistivity curve is drawn as an unbroken line. The resistivity model is displayed numerically as resistivity values (Ohmm) and layer thicknesses (m) and also as a histogram where the x-axis shows the depth and the y-axis the resistivity values. The value of the chi-square sum is also displayed on the plot. The plot is marked by a station identification which is identical to that part of the output plot file name which is in front of the point (.). If e.g. the output plot file is given the name HE105.PLT, then the plot will be marked as STATION: HE105.

To plot the results on the terminal, simply type SPLOT followed by the name of the output plot file and press return. This initiates a command procedure that appends the plot file to the grid file RES.GRD and plots the results on the screen by the program VIEW.EXE. The plot can be zoomed in by striking the + key and shifted both in horizontal and vertical directions by striking the arrow keys. To return to the DOS prompt, strike the Esc-key, then q and return.

To plot a hard copy on a printer or a plotter, simply type PPLOT followed by the name of the output plot file and return. This initiates a command procedure that appends the plot file to the grid file and plots the results by the program PLOT.EXE. The plot program asks if the plot is to be shifted, and if confirmed, how much in each direction. The first time a hard copy is produced or if the output device is changed, it may be necessary to reset the output device specification. This is done by changing the working directory to the directory containing the program PLOT.EXE and typing PLOT/I. The plot program displays the current output device specification and asks if it is to be changed. If this is answered positively, it displays a list of possible output devices and the appropriate choice can be made and saved by following a step by step procedure conducted by the plot program.

An example of an output plot, plotted on a printer, is shown in Figure 4.1.

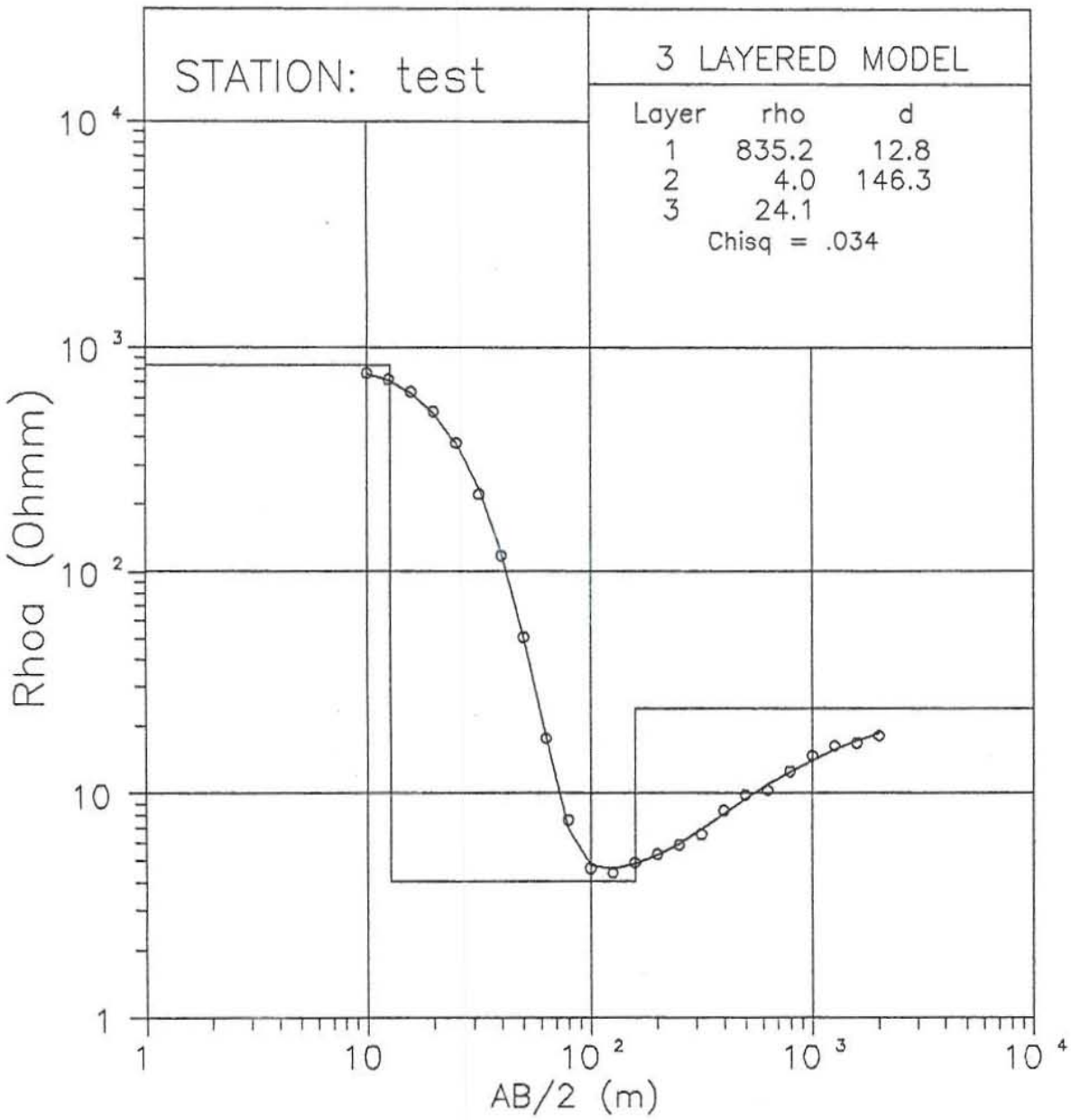

Figure 4.1 An output plot from SLINV.

5. REFERENCES

Johansen, H. K., 1975: An interactive computer/graphic-display-terminal system for interpretation of resistivity soundings. Geophysical Prospecting 23, pp. 449-458.

Johansen, H. K., 1977: A man/computer interpretation system for resistivity soundings over a horizontally stratified earth. Geophysical Prospecting 25, pp. 667-691.

APPENDIX

List of source files

```
*************************************************************************
*                                                                       *
*                              SLINV                                    *
*                                                                       *
*       This is a non-linear least-square inversion program for         *
*       inversion of Schlumberger resistivity soundings.                *
*       The program uses an iterative Levenberg-Marquardt inversion     *
*       algorithm described by H. K. Johansen (1977) together with      *
*       a forward routine based on the linear filter method as          *
*       described by H. K. Johansen (1975).                             *
*                                                                       *
*       *       *       *       *       *       *       *       *     *
*                                                                       *
*       Copyright (C) UNU Reykjavík Iceland 1988.                       *
*                                                                       *
*       This program was written for the United Nations University      *
*       Geothermal Training Programme in Reykjavík, Iceland.            *
*                                                                       *
*       Author: Knútur Árnason, National Energy Authority of Iceland,   *
*       Geothermal Division.                                            *
*                                                                       *
*       *       *       *       *       *       *       *       *     *
*                                                                       *
*       The following subroutines are called by the main program:       *
*                                                                       *
*       RDAT    opens and reads data (X,YM) from an input file and      *
*               computes weight coefficients SIG. It also reads a       *
*               starting model (P) from the terminal.                   *
*       FLT     returns the digital filter, C, used in SLFW and SLDR.   *
*       SLFW    calculates ordinant values (YC) from a given model (P). *
*       SLDR    calculates the partial derivative matrix Aij=dYi/dPj.   *
*       CHSQ    calculates the chi-square sum,                          *
*               CHI=sqrt(sum(((YM-YC)/SIG)**2).                         *
*       SVDC    does singular value decomposition on the matrix A,      *
*               A=U*W*V**t, where U is the data eigenvector matrix and  *
*               replaces A on output, W is a diagonal matrix (stored    *
*               as a vector) containing the eigen-(singular)values and  *
*               V is the parameter eigenvector matrix.                  *
*       ORDW    orders the eigenvalues in W in an increasing order and  *
*               returns in WR.                                          *
*       NEWP    calculates increments (PT) that are to be added to      *
*               the model parameter (P) in order to decrease the        *
*               chi-square sum (CHI).                                   *
*       WROT    writes out the final model parameters (P) and the       *
*               measured and calculated apparent resistivity values     *
*               (YM and YC) both into an output plot file and an        *
*               output list file. It also writes the data and           *
*               parameter eigenvectors, the eigenvalues and the         *
*               correlation matrix into the output list file.           *
*                                                                       *
*       The following vectors and matrices are used:                    *
*                                                                       *
*       X(NDM)          abscissa values ln(AB/2) (from input file).     *
*       YM(NDM)         ordinant values ln(Rhoam) (from input file).    *
*       SIG(NDM)        weight parameters for the YM values.            *
*                       The higher the weight parameter the smaller     *
*                       the contribution to the chi-square sum.         *
*       P(NPM)          model parameters.                               *
*       IPF(NPM)        array telling which parameters are fixed        *
*                       (IPF>0) and which are not fixed (IPF=0).        *
*       FIX(NPM)        array of characters; '*' for fixed parameters   *
*                       and ' ' for non-fixed parameters.               *
*       YC(NDM)         calculated ordinant values ln(Rhoac)            *
*                       from the model parameters P.                    *
*       PN(NPM)         increments to be added to the model parameters. *
```

```
*       PT(NPM)         temporary model parameters.                     *
*       YT(NDM)         temporary ordinant values (calculated from PT). *
*       A(NDM,NPM)      partial derivative matrix on output from SLDR   *
*                       but data eigenvector matrix on output from SVDC.*
*       W(NPM)          eigenvalues.                                    *
*       WR(NPM)         eigenvalues ordered in an increasing order.     *
*       V(NPM,NPM)      parameter eigenvector matrix.                   *
*       C(141)          digital filter coefficients used in SLFW        *
*                       and SLDR.                                       *
*                                                                       *
*       The following parameters are used in the main program:          *
*                                                                       *
*       NDM     maximum number of data points (X,YM).                   *
*       NPM     maximum number of model parameters (P).                 *
*       ND      number of data points (X,YM).                           *
*       NP      number of model parameters (P).                         *
*       NF      number of non-fixed model parameters.                   *
*       WMOD    weighing mode (RE or AB).                               *
*       RW      weight parameter.                                       *
*       CHI     chi-square sum.                                         *
*       CHIT    temporary chi-square sum.                               *
*       DCHI    fractional decrease of chi-square sum.                  *
*       CHIS    stop-check parameter for CHI (stop if CHI<CHIS).        *
*       DCHIS   stop-check parameter for DCHI (stop if DCHI<DCHIS).     *
*       ITR     iteration counter.                                      *
*       ITRS    stop-check parameter for ITR (stop if ITR<ITRS).        *
*       XLA     Marquardt parameter.                                    *
*       ILA     retry counter which is an index for XLA (XLA=WR(ILA)).  *
*       ISTOP   stop index telling on which stop-check the program      *
*               stopped ( CHI<CHIS => ISTOP=0, DCHI<DCHIS => ISTOP=1,   *
*               ITR>ITRS => ISTOP=2, ILA>NP => ISTOP=3 ).               *
*                                                                       *
*       *       *       *       *       *       *       *       *     *
*                                                                       *
*       References:                                                     *
*                                                                       *
*       Johansen, H. K., 1975: An interactive computer/graphic-display- *
*           terminal system for interpretation of resistivity           *
*           soundings, Geophysical Prospecting 23, pp. 449-458.         *
*       Johansen, H. K., 1977: A man/computer interpretation system     *
*           for resistivity soundings over a horizontally stratified    *
*           earth, Geophysical Prospecting 25, pp. 667-691.             *
*                                                                       *
************************************************************************

      PROGRAM SLINV

      PARAMETER (NDM=51,NPM=50)
      COMMON C(141)

      CHARACTER*12 OUTF,PLTF
      CHARACTER*1 FIX(NPM),ANSIT
      CHARACTER*2 WMOD

      DIMENSION X(NDM),YM(NDM),YC(NDM),YT(NDM),SIG(NDM)
      DIMENSION P(NPM),PN(NPM),PT(NPM),W(NPM),WR(NPM),IPF(NPM)
      DIMENSION A(NDM,NPM),V(NPM,NPM)

C     Read abscissas (X) and ordinants (YM) and compute weight
C     parameters (SIG) and read initial model parameters.

      CALL RDAT (X,YM,SIG,IMA,ND,P,IPF,NP,NF,NDM,NPM,FIX,WMOD,RW)

      WRITE (*,'(A,$)') ' NUMBER OF ITERATIONS: '
      READ (*,'(I2)') ITRS
      WRITE (*,'(/)')
```

```
C       Read output file name from terminal and
C       open output file as logical unit 1.

        WRITE (*,'(A,$)') ' OUTPUT LIST FILE: '
        READ (*,'(A)') OUTF
        OPEN (UNIT=1,FILE=OUTF)

C       Read output plotfile name from terminal

        WRITE (*,'(A,$)') ' OUTPUT PLOT FILE: '
        READ (*,'(A)') PLTF
        WRITE (*,'(/)')
        WRITE (*,'(A)') ' ** WORKING **'
        NL=(NP+1)/2

C       Write weighing mode and initial model in output file.

        WRITE (1,'(/)')
        WRITE (1,'(A,A,A,F3.1)') ' WEIGHING MODE = ',WMOD,
     &  '    WEIGTH PARAMETER RW = ',RW
        WRITE (1,'(/)')
        WRITE (1,'(A)') ' INITIAL MODEL PARAMETERS:'
        WRITE (1,'(/)')
        WRITE (1,'(A,8(F8.3,A1))') ' rho:',(EXP(P(I)),FIX(I),I=1,NL)
        WRITE (1,'(A,8(F8.3,A1))') '   d:',(EXP(P(I)),FIX(I),I=NL+1,NP)

C       The following two parameters are stop check parameters
C       for the iteration process.

        CHIS=1.0E-03
        DCHIS=1.0E-05

C       Get the digital filter C (transfered through common).

        CALL FLT

C       Initialize the iteration counter.

        ITR=0

C       Compute ordinants (YC) from the initial model parameters (P).

        CALL SLFW (YC,P,IMA,ND,NP,NDM,NPM)

C       Calculate the chi-square sum.

        CALL CHSQ (YM,YC,SIG,ND,CHI,NDM)

        write (*,'(a,e11.5)') ' CHI=',chi
        WRITE (1,'(A,E11.5)') '  CHI=',CHI
        WRITE (1,'(/)')

C       Here starts the iteration process.

        DCHI=1.0

  1             ITR=ITR+1

        write (*,'(/)')
        write (*,'(a,i3)') ' ITR=',itr

C               Calculate the partial derivative matrix A.

        write (*,'(a)') ' ** WORKING **'
```

```
                CALL SLDR(YC,P,IPF,ND,NP,IMA,A,NDM,NPM)

C               Do a singular value decomposition on A, A=U*W*V**t.
C               On input A is the partial derivative matrix,
C               on output A is the data eigenvector matrix U.

                CALL SVDC (A,ND,NF,NDM,NPM,W,V)

C               Order the eigenvalues W in an increasing order into WR
C               to be used as Marquardt parameters.

                CALL ORDW (W,WR,NF,NPM)

C               Initialize the Marquardt index to start with the smallest
C               Marquardt parameter.

                ILA=1

C               Here starts a loop that increases the Marquardt
C               parameter as long as the temporary chi-square sum,
C               CHIT, is not less than the chi-square sum, CHI.


  2                     XLA=DCHI**(1./REAL(ILA+1))*WR(ILA)
                        IF (XLA.LT.0.01) THEN
                          ILA=ILA+1
                          IF (ILA.GT.NF) THEN
                            ISTOP=3
                            GOTO 3
                          ENDIF
                          GOTO 2
                        ENDIF

        write (*,'(a,i2,a,e11.4)') '     ILA=',ila,'   XLA=',xla

C                       Calculate increments, PN, to be added to
C                       the model parameters, P.

                        CALL NEWP (YM,YC,SIG,A,W,V,XLA,PN,ND,NF,NDM,NPM)

C                       Check if the increments in log-parameters, PN,
C                       change the model parameters by more than
C                       factor 50 and then damp PN by increasing ILA.

                        DO 10 J=1,NF
                          IF (ABS(PN(J)).GT.3.9) THEN
                            ILA=ILA+1
                            IF (ILA.GT.NF) THEN
                              ISTOP=3
                              GOTO 3
                            ENDIF
                            GOTO 2
                          ENDIF
  10                    CONTINUE

C                       Add the increments, PN, to the parameter
C                       vector, P, to get new temporary parameter
C                       vector, PT.

                        I1=1
                        DO 11 I=1,NP
                          IF (IPF(I).EQ.0) THEN
                            PT(I)=P(I)+PN(I1)
                            I1=I1+1
                          ELSE
                            PT(I)=P(I)
```

```
                        ENDIF
   11                 CONTINUE
   41                 CONTINUE

C       Write on the terminal the best model obtained so far
C       and the temporary model to be tried next.

        write (*,'(a,8(f8.2,a1))') '  rho:',(exp(p(i)),fix(i),i=1,nl)
        write (*,'(a,8(f8.2,a1))') '    d:',(exp(p(i)),fix(i),i=nl+1,np)
        write (*,'(a,8(f8.2,a1))') ' rhot:',(exp(pt(i)),fix(i),i=1,nl)
        write (*,'(a,8(f8.2,a1))') '   dt:',(exp(pt(i)),fix(i),i=nl+1,np)

C                       Calculate temporary ordinant values, YT,
C                       from the temporary model PT.

                        CALL SLFW (YT,PT,IMA,ND,NP,NDM,NPM)

C                       Calculate the temporary chi-square sum, CHIT.

                        CALL CHSQ (YM,YT,SIG,ND,CHIT,NDM)

        write (*,'(a,e11.5)') '  CHI=',chi
        write (*,'(a,e11.5)') ' CHIT=',chit

C                       Check if CHIT is less than CHI.
C                       If not then increase the Marquardt
C                       parameter XLA and try again as long
C                       as ILA is not greater than NF.

                        IF (CHIT.GE.CHI) THEN
                          IF (ILA.GE.NF) THEN
                            ISTOP=3
                            GOTO 3
                          ENDIF
                          ILA=ILA+1
                          GOTO 2
                        ENDIF

C               For CHIT less than CHI keep the temporary
C               values and calculate the fractional decrease,
C               DCHI, of the chi-square sum.

                DO 12 I=1,ND
                  YC(I)=YT(I)
   12           CONTINUE

                DO 13 J=1,NP
                  P(J)=PT(J)
   13           CONTINUE
                DCHI=(CHI-CHIT)/CHI
                CHI=CHIT

C               Check if CHI is less than CHIS,
C               then stop the iteration.

                IF (CHI.LE.CHIS) THEN
                  ISTOP=0
                  GOTO 3
                ENDIF

C               Check if the fractional decrease DCHI is
C               less than DCHIS, then stop the iteration.

                IF (DCHI.LT.DCHIS) THEN
                  ISTOP=1
                  GOTO 3
```

```
                ENDIF

C               Check if the number of iterations, ITR, is less
C               than ITRS, then write the best model so far in the
C               output file and continue the iteration process.

                IF (ITR.LT.ITRS) THEN
  4               CONTINUE
                  WRITE (1,100) ' ITR=',ITR,'  ILA=',ILA,'  CHI=',
     &            CHI,'  DCHI=',DCHI
 100              FORMAT (A,I2,A,I2,A,E10.4,A,E10.4)
                  WRITE (1,'(A,8(F8.2,A1))') ' rho:',
     &            (EXP(P(I)),FIX(I),I=1,NL)
                  WRITE (1,'(A,8(F8.2,A1))') '   d:',
     &            (EXP(P(I)),FIX(I),I=NL+1,NP)
                  WRITE (1,'(/)')
                  GOTO 1
                ENDIF

C               If ITR is equal to ITRS, then ask if more iterations
C               are to be performed or to stop.

                WRITE (*,'(/)')
                WRITE (*,'(A,I2,A,$)') ' FINISHED ',ITR,' ITERATIONS,
     &          WANT MORE ? (Y/N) :'
                READ (*,'(A)') ANSIT
                IF (ANSIT.EQ.'Y') THEN
                  WRITE (*,'(A,$)') ' HOW MANY MORE ? :'
                  READ (*,*) ITRS1
                  ITRS=ITRS+ITRS1
                  GOTO 4
                ELSEIF (ANSIT.EQ.'y') THEN
                  WRITE (*,'(A,$)') ' HOW MANY MORE ? :'
                  READ (*,*) ITRS1
                  ITRS=ITRS+ITRS1
                  GOTO 4
                ELSE
                  ISTOP=2
                ENDIF

  3     CONTINUE

C       Write out results in output list file and output plot file.

        CALL WROT (X,YM,YC,SIG,P,CHI,ISTOP,ITR,ND,NP,NF,NDM,NPM,FIX,
     &  PLTF,A,W,V)

        WRITE (*,'(/)')

        STOP
        END
```

```
      SUBROUTINE RDAT (X,YM,SIG,IMA,ND,P,IPF,NP,NF,NDM,NPM,FIX,
     &  WMOD,RW)

C*********************************************************************
C                                                                     *
C                              RDAT                                   *
C                                                                     *
C     This routine reads input data from file. The natural            *
C     logarithms of the AB/2 values (abscissas) are read and stored   *
C     in X1, the natural logarithms of the measured apparent          *
C     resistivity values (ordinants) are read and stored in YM1.      *
C     Interpolated data points evenly distributed on log-scale with   *
C     10 points per decade are stored in X and YM. Calculated         *
C     weight factors for each data pair (X,YM) are stored in SIG.     *
C     ND is the number of data triplets (X,YM,SIG) and must be <42.   *
C     The routine also reads initial model parameters, stored in P,   *
C     from the terminal. NP is the number of model parameters and     *
C     must be <19. For layered earth models P(1),...,P((NP+1)/2)      *
C     are resistivity values for the NL=(NP+1)/2 layers and           *
C     P(NL),...,P(NP) are the layer thicknesses.                      *
C                                                                     *
C*********************************************************************

      DIMENSION X1(51),YM1(51),X(NDM),YM(NDM)
      DIMENSION SIG(NDM),P(NPM),IPF(NPM)
      CHARACTER*12 FNAME
      CHARACTER*1 FIX(NPM)
      CHARACTER*2 WMOD

C     Read input filename from terminal and open input file.

      WRITE (*,'(/)')
      WRITE (*,'(A,$)') ' INPUT FILE:'
      READ (*,'(A)') FNAME
      OPEN (UNIT=1,FILE=FNAME,STATUS='OLD')

C     Read instructions for how to weight the data.

      WRITE (*,'(/)')
      WRITE (*,'(A,$)') ' WEIGHT WITH RESPECT TO RESISTIVITY OR'
      WRITE (*,'(A,$)') ' AB/2 ? (RE/AB): '
      READ (*,'(A)') WMOD
      WRITE (*,'(A,$)') ' WEIGHT PARAMETER (-1<=RW<=1): '
      READ (*,*) RW

C     Read data from input file

      DO 11 I=1,NDM+1
        READ (1,*,END=12) X1(I),YM1(I)
        YM1(I)=ALOG(YM1(I))
11    CONTINUE

12    ND1=I-1
      CLOSE (UNIT=1)
      DX=ALOG(10.)/10.

C     Interpolate between the measured data points in
C     order to get data equally distributed on log-scale
C     with 10 points per decade and calculate weight factor
C     for each data point.

      IMIN=INT(LOG(X1(1))/DX-0.25)
      IMAX=INT(LOG(X1(ND1))/DX+0.25)
      ND=IMAX-IMIN
```

```
      I1=1

      DO 10 I=1,ND
        X(I)=EXP((IMIN+I)*DX)
        IF (X(I).GT.X1(I1+1)) I1=MIN(I1+1,ND1-1)
        YM(I)=YM1(I1)+(YM1(I1+1)-YM1(I1))/LOG(X1(I1+1)/X1(I1))
     &    *LOG(X(I)/X1(I1))
        IF (WMOD.EQ.'AB') THEN
          SUM=SUM+((IMIN+I)*DX)**(2.*RW)
        ELSEIF (WMOD.EQ.'ab') THEN
          WMOD='AB'
          SUM=SUM+((IMIN+I)*DX)**(2.*RW)
        ELSE
          WMOD='RE'
          SUM=SUM+YM(I)**(2.*RW)
        ENDIF
 10   CONTINUE

      SUM=SQRT(SUM/ND)

      DO 15 I=1,ND
        IF (WMOD.EQ.'AB') THEN
          SIG(I)=((IMIN+I)*DX)**(-RW)*SUM
        ELSEIF (WMOD.EQ.'ab') THEN
          SIG(I)=((IMIN+I)*DX)**(-RW)*SUM
        ELSE
          SIG(I)=YM(I)**(-RW)*SUM
        ENDIF
 15   CONTINUE

      IMA=IMAX

C     Read initial model parameters from the terminal.

      WRITE (*,'(/)')
      WRITE (*,'(A)') ' INITIAL MODEL PARAMETERS:'
      WRITE (*,'(/)')
      WRITE (*,'(A,$)') ' NUMBER OF LAYERS:'
      READ (*,'(I2)') NL
      NF=0
      WRITE (*,'(/)')
      WRITE (*,'(A,$)') ' TYPE RESISTIVITY VALUES AND'
      WRITE (*,'(A)') ' LAYER THICKNESSES,* IF TO BE FIXED:'

      DO 13 I=1,NL-1
        IPF(I)=0
        WRITE (*,'(/)')
        WRITE (*,'(A,I1,A,$)') ' rho(',I,'): '
        READ (*,'(F10.0,A)') P(I),FIX(I)
        P(I)=ALOG(P(I))
        IF (FIX(I).EQ.'*') THEN
          IPF(I)=I
          NF=NF+1
        ENDIF
        IPF(NL+I)=0
        WRITE (*,'(A,I1,A,$)') '   d(',I,'): '
        READ (*,'(F10.0,A)') P(NL+I),FIX(NL+I)
        P(NL+I)=ALOG(P(NL+I))
        IF (FIX(NL+I).EQ.'*') THEN
          IPF(NL+I)=NL+I
          NF=NF+1
        ENDIF
 13   CONTINUE

      IPF(NL)=0
      WRITE (*,'(/)')
```

```
      WRITE (*,'(A,I1,A,$)') ' rho(',NL,'): '
      READ (*,'(F10.0,A)') P(NL),FIX(NL)
      P(NL)=ALOG(P(NL))
      IF (FIX(NL).EQ.'*') THEN
        IPF(NL)=NL
        NF=NF+1
      ENDIF
      WRITE (*,'(/)')

C     Compute the number of parameters in the model, NP, and
C     the number of free parameters, NF.

      NP=2*NL-1
      NF=NP-NF
      RETURN
      END
```

```
      SUBROUTINE FLT

C*************************************************************************
C                                                                        *
C                              FLT.                                      *
C                                                                        *
C     This routine stores and returns the 141 point J1 digital           *
C     filter used in SLFW and SLDR to calculate apparent resis-          *
C     tivity and partial derivative matrix.                              *
C                                                                        *
C*************************************************************************

      COMMON C(141)
      DIMENSION CC(141)

C     Scale the filter coefficients.

      DO 1 L=1,141
        C(L)=CC(L)*1.0E-08
  1   CONTINUE

C     Here CC=C*1.0E+08 are stored as data (C are the digital filter
C     coefficients).

      DATA CC /6174.,-12484.,12726.,-12975.,13231.,-13494.,13765.
     &  ,-14043.,14330.,-14625.,14930.,-15244.,15567.,-15901.,16246.
     &  ,-16602.,16971.,-17352.,17746.,-18154.,18577.,-19015.,19469.
     &  ,-19941.,20429.,-20936.,21463.,-22009.,22577.,-23166.,23779.
     &  ,-24416.,25079.,-25768.,26487.,-27235.,28016.,-28830.,29680.
     &  ,-30568.,31496.,-32467.,33484.,-34549.,35666.,-36838.,38069.
     &  ,-39363.,40724.,-42156.,43666.,-45259.,46940.,-48717.,50596.
     &  ,-52587.,54697.,-56936.,59314.,-61845.,64540.,-67414.,70484.
     &  ,-73767.,77284.,-81057.,85111.,-89475.,94183.,-99267.,104775.
     &  ,-110741.,117248.,-124303.,132085.,-140461.,149959.,-159826.
     &  ,171917.,-182946.,199955.,-209469.,239052.,-234543.,304916.
     &  ,-234124.,453990.,-106745.,899282.,550573.,2442523.,3250077.
     &  ,7926675.,13023345.,25610307.,41150741.,64231809.,72803988.
     &  ,36118538.,-100406442.,-242172543.,20052460.,444506381.
     &  ,-489348908.,294899398.,-137791072.,61285163.,-29362551.
     &  ,15817356.,-9504597.,6226174.,-4353505.,3198475.,-2441493.
     &  ,1920840.,-1548505.,1273595.,-1065148.,903512.,-775750.,673079.
     &  ,-589375.,520264.,-462558.,413891.,-372478.,336951.,-306251.
     &  ,279543.,-256168.,235594.,-217394.,201216.,-186773.,173826.
     &  ,-162176.,151657.,-142126.,133463.,-125568.,60905./

      RETURN
      END
```

```
      SUBROUTINE SLFW (Y,P,IMA,ND,NP,NDM,NPM)

*************************************************************************
C                                                                       *
C                              SLFW                                     *
C                                                                       *
C     This is a forward routine that calculates apparent resistivity   *
C     curve for a given model rho(1),..,rho(NL),d(1),..,d(NL-1)         *
C     stored in the vector P. NL is the number of layers and must be    *
C     NL<11. The apparent resistivity, stored in the vector Y, is       *
C     computed as a function of AB/2, stored in the vector X, by        *
C     using the gradient approximation and the digital (J1) filter      *
C     from H.K. Johansen (1975). The resistivity transform, stored      *
C     in T, is convolved with the 141 point digital filter,             *
C     stored in C. The AB/2 values are equally distributed on           *
C     log-scale with 10 points per decade.                              *
C                                                                       *
C*************************************************************************

      COMMON C(141)

      DIMENSION Y(NDM),T(191),P(NPM)

C     Change the model parameters from logarithmic to linear form.

      DO 12 J=1,NP
        P(J)=EXP(P(J))
 12   CONTINUE

      DX=0.2302585

C     Setting up the resistivity transform T(L).

      NL=(NP+1)/2
      S=-1.7239458
      RX=EXP(DX)
      SL=EXP(S+DX*(IMA+101))
      RK=(P(NL-1)-P(NL))/(P(NL-1)+P(NL))

      DO 14 L=80,1,-1
        SLA=(RX**L)/SL
        REX=2.*P(2*NL-1)*SLA
        AEXP=EXP(-REX)
        T(L)=P(NL-1)*(1.0-RK*AEXP)/(1.0+RK*AEXP)

        DO 13 J=NL-2,1,-1
          REX=2.*P(NL+J)*SLA
          AEXP=EXP(-REX)
          WD=(1.0-AEXP)/(1.0+AEXP)
          T(L)=(P(J)*WD+T(L))/(1.0+WD*T(L)/P(J))
 13     CONTINUE

        IF (ABS(T(L)-P(NL)).LE.1.0E-02) THEN
          LMI=L
          GOTO 15
        ENDIF

 14   CONTINUE

 15   CONTINUE

      DO 17 L=81,140+ND
        SLA=(RX**L)/SL
        REX=2.*P(2*NL-1)*SLA
        AEXP=EXP(-REX)
```

```
          T(L)=P(NL-1)*(1.0-RK*AEXP)/(1.0+RK*AEXP)

          DO 16 J=NL-2,1,-1
            REX=2.*P(NL+J)*SLA
            AEXP=EXP(-REX)
            WD=(1.0-AEXP)/(1.0+AEXP)
            T(L)=(P(J)*WD+T(L))/(1.0+WD*T(L)/P(J))
 16       CONTINUE

          IF (ABS(T(L)-P(1)).LE.1.0E-02) THEN
            LMA=L
            GOTO 18
          ENDIF
 17     CONTINUE

 18     CONTINUE
        S1=0.0

        IF (LMI.GT.ND) THEN

          DO 19 L=1,LMI-ND
            S1=S1+C(L)
 19       CONTINUE

        ENDIF

        S2=0.0

        IF ((LMA+1-ND).LE.141) THEN

          DO 20 L=LMA+1-ND,141
            S2=S2+C(L)
 20       CONTINUE

        ENDIF

C       Convolve the resistivity transform T with the filter C.

        DO 22 I=1,ND
          Y(I)=0.0

          IF ((LMI-ND+I).GT.0) THEN
            S1=S1+C(LMI-ND+I)
          ENDIF

          L1=LMI+1+I-ND

          IF (L1.LT.1) THEN
            L1=1
          ENDIF

          L2=LMA-1+I-ND

          IF (L2.GT.141) THEN
            L2=141
          ENDIF

          DO 21 L=L1,L2
            Y(I)=Y(I)+T(L-I+ND)*C(L)
 21       CONTINUE

          Y(I)=Y(I)+P(NL)*S1+P(1)*S2

          IF ((LMA+I-ND).LE.141) THEN
            S2=S2-C(LMA+I-ND)
          ENDIF
```

```
   22    CONTINUE

C        Change the calculated apparent resistivities to logarithmic form.

         DO 23 I=1,ND
           Y(I)=ALOG(Y(I))
   23    CONTINUE

C        Change the model parameters back to logarithmic form.

         DO 24 J=1,NP
           P(J)=ALOG(P(J))
   24    CONTINUE

         RETURN
         END
```

```
      SUBROUTINE SLDR (Y,P,IPF,ND,NP,IMA,A,NDM,NPM)

C*********************************************************************
C                                                                     *
C                                SLDR                                 *
C                                                                     *
C     This is a routine that calculates the partial derivative matrix *
C     Aij=dln(Yi)/dln(Pj) for layered resistivity structure.          *
C     The layered model rho(1),..,rho(NL),d(1),..,d(NL-1) is          *
C     stored in the vector P. NL is the number of layers and must be  *
C     NL<11, ND is the number of abscissa values, NP=2*NL-1 is the    *
C     number of model parameters and IMA=ln(Xmax)/10 is the maximum   *
C     coordinant number of abscissas. TB, WB and TRD are temporary    *
C     arrays.                                                         *
C                                                                     *
C*********************************************************************

      COMMON C(141)

      PARAMETER (NTM=38)
      DIMENSION Y(NDM),P(NPM),IPF(NPM),TB(NTM),WB(NTM)
      DIMENSION TRD(191,NTM),A(NDM,NPM)

C     Change the incoming logarithmic quantities
C     to non-logarithmic quantities.

      DO 11 I=1,ND
        Y(I)=EXP(Y(I))
 11   CONTINUE

      DO 12 J=1,NP
        P(J)=EXP(P(J))
 12   CONTINUE

      NL=(NP+1)/2
      DX=0.2302585
      S=-1.7239458
      RX=EXP(DX)

C     Set up a kernel matrix, T(L,J), which convolved with
C     the filter gives the partial derivative matrix.

      SL=EXP(S+DX*(IMA+101))
      RK=(P(NL-1)-P(NL))/(P(NL-1)+P(NL))

      DO 15 L=1,140+ND
        SLA=(RX**L)/SL
        REX=2.*P(NP)*SLA
        AEXP1=EXP(-REX)
        T=(1.0-RK*AEXP1)/(1.0+RK*AEXP1)
        TT=1.0
        IF (NL.EQ.2) GOTO 1
        TM=T*P(NL-1)
        TB(NL-2)=TM/P(NL-2)

        DO 13 J=NL-2,1,-1
          REX=2.*P(NL+J)*SLA
          AEXP=EXP(-REX)
          WB(J)=(1.0-AEXP)/(1.0+AEXP)
 13     CONTINUE

        IF (NL.GT.3) THEN

          DO 31 J=NL-3,1,-1
            TM=(WB(J+1)*P(J+1)+TM)/(1.0+WB(J+1)*TM/P(J+1))
```

```
              TB(J)=TM/P(J)
   31         CONTINUE

            ENDIF

            DO 14 J=1,NL-2
              RN=1.+WB(J)*TB(J)
              TRD(L,J)=TT*(2.*RN-1.+TB(J)**2)*WB(J)/(RN**2)
          TRD(L,NL+J)=TT*(1.-TB(J)**2)*(1.-WB(J)**2)*P(J)*SLA/(RN**2)
              TT=TT*(1.0-WB(J)**2)/(RN**2)
   14         CONTINUE

   1          RR=4.*P(NL-1)*AEXP1/((1.+RK*AEXP1)**2)
              TRD(L,NL)=TT*RR*P(NL-1)/((P(NL-1)+P(NL))**2)
              TRD(L,NL-1)=TT*T-TRD(L,NL)*P(NL)/P(NL-1)
              TRD(L,NP)=TT*RR*RK*SLA
   15       CONTINUE

C           Now convolve with the filter and compute Aij.

            J1=1

            DO 20 J=1,NP

              IF (IPF(J).EQ.0) THEN

                DO 19 I=1,ND

                  A(I,J1)=0.0

                  DO 16 L=70,1,-1
                    IF (ABS(TRD(ND-I+L,J)).LT.1.E-08) GOTO 17
                    A(I,J1)=A(I,J1)+TRD(ND-I+L,J)*C(L)
   16             CONTINUE

   17             CONTINUE

                  DO 18 L=71,141
                    IF (ABS(TRD(ND-I+L,J)).LT.1.E-08) GOTO 2
                    A(I,J1)=A(I,J1)+TRD(ND-I+L,J)*C(L)
   18             CONTINUE

   2              A(I,J1)=A(I,J1)*P(J)/Y(I)
   19           CONTINUE

                J1=J1+1
              ENDIF

   20       CONTINUE

C           Change the apparent resistivities and the model parameters
C           back to logarithmic form.

            DO 21 I=1,ND
              Y(I)=ALOG(Y(I))
   21       CONTINUE

            DO 22 J=1,NP
              P(J)=ALOG(P(J))
   22       CONTINUE

            RETURN
            END
```

```
      SUBROUTINE CHSQ(YM,YC,SIG,ND,CHI,NDM)

C*************************************************************************
C                                                                        *
C                                  CHSQ                                  *
C                                                                        *
C     This routine computes the square root of the sum of                *
C     ((YM-YC)/SIG)**2 from 1 to ND and returns it as CHI.               *
C                                                                        *
C*************************************************************************

      DIMENSION YM(NDM),YC(NDM),SIG(NDM)

C     Calculate the chi-square sum.

      CHI=0.0

      DO 11 I=1,ND
       CHI=CHI+((YM(I)-YC(I))/SIG(I))**2
 11   CONTINUE

      CHI=SQRT(CHI/REAL(ND))

      RETURN
      END
```

```
      SUBROUTINE SVDC(A,M,N,NDM,NPM,W,V)

C*********************************************************************
C                                                                    *
C                              SVDC                                  *
C            Singular Value Decomposition Algorithm.                 *
C      Given a matrix A, with logical dimensions M and N (MxN) and   *
C      physical dimensions MP and NP, this routine computes its      *
C      singular value decomposition, A=U*W*V**T. The matrix U replaces *
C      A on output. The diagonal matrix W is output as a vector W.   *
C      The matrix V (not the transpose V**T) is output as V. M must  *
C      be greater or equal to N; if it is smaller, then A should be  *
C      filled up to square with zero rows.                           *
C      This routine is a slightly altered version of the routine     *
C      SVDCMP in the book: Numerical Recipes, The Art of Scientic    *
C      Computing by W. H. Press et.al., Cambridge Univ. Press 1986.  *
C                                                                    *
C*********************************************************************

      PARAMETER (NMAX=100)
      DIMENSION A(NDM,NPM),W(NPM),V(NPM,NPM),RV1(NMAX)
      IF (M.LT.N) PAUSE 'You must augment A with extra zero rows'

C     Householder reduction to bidiagonal form.

      G=0.0
      SCALE=0.0
      ANORM=0.0
      DO 25 I=1,N
        L=I+1
        RV1(I)=SCALE*G
        G=0.0
        S=0.0
        SCALE=0.0
        IF (I.LE.M) THEN
          DO 11 K=I,M
            SCALE=SCALE+ABS(A(K,I))
11        CONTINUE
          IF (SCALE.NE.0.0) THEN
            DO 12 K=I,M
              A(K,I)=A(K,I)/SCALE
              S=S+A(K,I)*A(K,I)
12          CONTINUE
            F=A(I,I)
            G=-SIGN(SQRT(S),F)
            H=F*G-S
            A(I,I)=F-G
            IF (I.NE.N) THEN
              DO 15 J=L,N
                S=0.0
                DO 13 K=I,M
                  S=S+A(K,I)*A(K,J)
13              CONTINUE
                F=S/H
                DO 14 K=I,M
                  A(K,J)=A(K,J)+F*A(K,I)
14              CONTINUE
15            CONTINUE
            ENDIF
            DO 16 K=I,M
              A(K,I)=SCALE*A(K,I)
16          CONTINUE
          ENDIF
        ENDIF
        W(I)=SCALE*G
```

```
          G=0.0
          S=0.0
          SCALE=0.0
          IF ((I.LE.M).AND.(I.NE.N)) THEN
            DO 17 K=L,N
              SCALE=SCALE+ABS(A(I,K))
 17         CONTINUE
            IF (SCALE.NE.0.0) THEN
              DO 18 K=L,N
                A(I,K)=A(I,K)/SCALE
                S=S+A(I,K)*A(I,K)
 18           CONTINUE
              F=A(I,L)
              G=-SIGN(SQRT(S),F)
              H=F*G-S
              A(I,L)=F-G
              DO 19 K=L,N
                RV1(K)=A(I,K)/H
 19           CONTINUE
              IF (I.NE.M) THEN
                DO 23 J=L,M
                  S=0.0
                  DO 21 K=L,N
                    S=S+A(J,K)*A(I,K)
 21               CONTINUE
                  DO 22 K=L,N
                    A(J,K)=A(J,K)+S*RV1(K)
 22               CONTINUE
 23             CONTINUE
              ENDIF
              DO 24 K=L,N
                A(I,K)=SCALE*A(I,K)
 24           CONTINUE
            ENDIF
          ENDIF
          ANORM=MAX(ANORM,(ABS(W(I))+ABS(RV1(I))))
 25     CONTINUE

C        Accumulation of right-hand transformations.

        DO 32 I=N,1,-1
          IF (I.LT.N) THEN
            IF (G.NE.0.0) THEN
              DO 26 J=L,N

C               Double division to avoid possible underflow:

                V(J,I)=(A(I,J)/A(I,L))/G
 26           CONTINUE
              DO 29 J=L,N
                S=0.0
                DO 27 K=L,N
                  S=S+A(I,K)*V(K,J)
 27             CONTINUE
                DO 28 K=L,N
                  V(K,J)=V(K,J)+S*V(K,I)
 28             CONTINUE
 29           CONTINUE
            ENDIF
            DO 31 J=L,N
              V(I,J)=0.0
              V(J,I)=0.0
 31         CONTINUE
          ENDIF
          V(I,I)=1.0
          G=RV1(I)
```

```
          L=I
   32   CONTINUE

C        Accumulation of left-hand transformations.

        DO 39 I=N,1,-1
          L=I+1
          G=W(I)
          IF (I.LT.N) THEN
            DO 33 J=L,N
              A(I,J)=0.0
   33       CONTINUE
          ENDIF
          IF (G.NE.0.0) THEN
            G=1.0/G
            IF (I.NE.N) THEN
              DO 36 J=L,N
                S=0.0
                DO 34 K=L,M
                  S=S+A(K,I)*A(K,J)
   34           CONTINUE
                F=(S/A(I,I))*G
                DO 35 K=I,M
                  A(K,J)=A(K,J)+F*A(K,I)
   35           CONTINUE
   36         CONTINUE
            ENDIF
            DO 37 J=I,M
              A(J,I)=A(J,I)*G
   37       CONTINUE
          ELSE
            DO 38 J=I,M
              A(J,I)=0.0
   38       CONTINUE
          ENDIF
          A(I,I)=A(I,I)+1.0
   39   CONTINUE

C        Diagonalization of the bidiagonal form.

        DO 49 K=N,1,-1

C        Loop over singular values.

          DO 48 ITS=1,30

C          Loop over allowed iterations.

            DO 41 L=K,1,-1
              NM=L-1
              IF ((ABS(RV1(L))+ANORM).EQ.ANORM) GO TO 2
              IF ((ABS(W(NM))+ANORM).EQ.ANORM) GO TO 1
   41       CONTINUE
    1       C=0.0
            S=1.0
            DO 43 I=L,K
              F=S*RV1(I)
              IF ((ABS(F)+ANORM).NE.ANORM) THEN
                G=W(I)
                H=SQRT(F*F+G*G)
                W(I)=H
                H=1.0/H
                C= (G*H)
                S=-(F*H)
                DO 42 J=1,M
                  Y=A(J,NM)
```

```
                Z=A(J,I)
                A(J,NM)=(Y*C)+(Z*S)
                A(J,I)=-(Y*S)+(Z*C)
 42           CONTINUE
            ENDIF
 43       CONTINUE
 2        Z=W(K)
          IF (L.EQ.K) THEN
C           Convergence.
            IF (Z.LT.0.0) THEN

C             Singular value is made non-negative.

              W(K)=-Z
              DO 44 J=1,N
                V(J,K)=-V(J,K)
 44           CONTINUE
            ENDIF
            GO TO 3
          ENDIF
          IF (ITS.EQ.30) PAUSE 'No convergence in 30 iterations'
          X=W(L)

C         Shift from bottom 2-by-2 minor:

          NM=K-1
          Y=W(NM)
          G=RV1(NM)
          H=RV1(K)
          F=((Y-Z)*(Y+Z)+(G-H)*(G+H))/(2.0*H*Y)
          G=SQRT(F*F+1.0)
          F=((X-Z)*(X+Z)+H*((Y/(F+SIGN(G,F)))-H))/X

C       Next QR transformation:

          C=1.0
          S=1.0
          DO 47 J=L,NM
            I=J+1
            G=RV1(I)
            Y=W(I)
            H=S*G
            G=C*G
            Z=SQRT(F*F+H*H)
            RV1(J)=Z
            C=F/Z
            S=H/Z
            F= (X*C)+(G*S)
            G=-(X*S)+(G*C)
            H=Y*S
            Y=Y*C
            DO 45 NM=1,M
              X=V(NM,J)
              Z=V(NM,I)
              V(NM,J)= (X*C)+(Z*S)
              V(NM,I)=-(X*S)+(Z*C)
 45           CONTINUE
            Z=SQRT(F*F+H*H)
            W(J)=Z

C           Rotation can be arbitrary if Z=0.

            IF (Z.NE.0.0) THEN
              Z=1.0/Z
              C=F*Z
              S=H*Z
```

```
                ENDIF
                F= (C*G)+(S*Y)
                X=-(S*G)+(C*Y)
                DO 46 NM=1,M
                  Y=A(NM,J)
                  Z=A(NM,I)
                  A(NM,J)= (Y*C)+(Z*S)
                  A(NM,I)=-(Y*S)+(Z*C)
46              CONTINUE
47            CONTINUE
              RV1(L)=0.0
              RV1(K)=F
              W(K)=X
48          CONTINUE
3           CONTINUE
49        CONTINUE
          RETURN
          END
```

```
      SUBROUTINE ORDW(W,WR,NF,NPM)

*************************************************************************
*                                                                       *
*                                ORDW                                   *
*                                                                       *
*       This routine orders NF array elements, stored in W, and         *
*       returns them in an increasing order in the array WR.            *
*                                                                       *
*************************************************************************

      DIMENSION W(NPM),WR(NPM)

C     Initialize WR.

      DO 11 I=1,NF
        WR(I)=W(I)
 11   CONTINUE

C     Reorder WR in an increasing order.

      DO 13 I=1,NF
        WMAX=WR(1)
        IT=1

        DO 12 J=1,NF+1-I
          IF (WR(J).GT.WMAX) THEN
            WMAX=WR(J)
            IT=J
          ENDIF
 12       CONTINUE

        WR(IT)=WR(NF+1-I)
        WR(NF+1-I)=WMAX
 13   CONTINUE

      RETURN
      END
```

```
      SUBROUTINE NEWP(YM,YC,SIG,A,W,V,XLA,PN,ND,NF,NDM,NPM)
*************************************************************************
*                                                                       *
*                              NEWP                                     *
*                                                                       *
*     This routine takes in measured (YM) and calculated (YC)           *
*     ordinants, weight parameters (SIG), Marquardt parameter           *
*     (XLA) and the partial derivative matrix dYMi/dPj, singular        *
*     value decomposed into dYMi/dPj=(A*W*V**t)i,j. The matrix          *
*     A contains the data eigenvectors as columns and V the             *
*     parameter eigenvectors as columns. The matrix W is diagonal       *
*     and contains the singular(eigen) values (and is stored as         *
*     a vector). The routine returns a vector (PN) containing           *
*     increments that are to be added to the previous model             *
*     parameter vector to get new and (hopefully) improved model.       *
*     The increment vector (PN) is calculated by an (Marquardt)         *
*     algorithm given by H. K. Johansen (1977).                         *
*     (ND and NF are the number of data (ordinant) values and free      *
*     model parameters, respectively)                                   *
*                                                                       *
*************************************************************************
      DIMENSION YM(NDM),YC(NDM),SIG(NDM),PN(NPM)
      DIMENSION A(NDM,NPM),W(NPM),V(NPM,NPM)

      DO 13 I=1,NF
        PN(I)=0.0

        DO 12 J=1,NF
          B=0.0

          DO 11 K=1,ND
            B=B+((YM(K)-YC(K))/SIG(K))*A(K,J)
 11       CONTINUE

          PN(I)=PN(I)+B*(W(J)/(W(J)**2+XLA**2))*V(I,J)
 12     CONTINUE

 13   CONTINUE

      RETURN
      END
```

```
      SUBROUTINE WROT(X,YM,YC,SIG,P,CHI,ISTOP,ITR,ND,NP,NF,NDM,NPM,
     &  FIX,PLTF,A,W,V)

*************************************************************************
*                                                                       *
*                              WROT                                     *
*                                                                       *
*     This routine writes out results from the inversion program        *
*     SLINV into an output list file that has been opened in the        *
*     main program as logical UNIT=1. It writes out the data            *
*     points (X,YM), the weight coefficients (1/SIG), the               *
*     final model parameters (P) and the corresponding calculated       *
*     ordinant values (YC). It writes the final chi-square sum (CHI),   *
*     the number of iterations performed (ITR), the stop-check          *
*     parameter (ISTOP), the data eigenvector matrix (A), the           *
*     parameter eigenvector matrix (V), the parameter eigenvalues       *
*     (W) and the correlation matrix.                                   *
*     The routine also writes the measured and calculated apparent      *
*     resistivity values, the final model and the chi-square sum        *
*     into a plot file that can be plotted either on the screen         *
*     or as a hard copy on a printer or a plotter.                      *
*                                                                       *
*************************************************************************

      DIMENSION X(NDM),YM(NDM),YC(NDM),SIG(NDM),P(NPM)
      DIMENSION A(NDM,NPM),W(NPM),V(NPM,NPM)

      CHARACTER*12 PLTF
      CHARACTER*8 STATION
      CHARACTER*1 FIX(NPM)

C     Write in the output list file on which stop check the
C     iteration terminated.

      WRITE (1,'(A,I2,A)') ' *** THE PROGRAM TERMINATED  AFTER '
     &  ,ITR,' ITRERATIONS ***'
      WRITE (1,'(/)')
      IF (ISTOP.EQ.0) THEN
        WRITE (1,'(A,I1,A)') ' ISTOP=',ISTOP,' * CHI CONVERGENCE *'
      ENDIF
      IF (ISTOP.EQ.1) THEN
        WRITE (1,'(A,I1,A)') ' ISTOP=',ISTOP,' * DCHI CONVERGENCE *'
      ENDIF
      IF (ISTOP.EQ.2) THEN
        WRITE (1,'(A,I1,A)') ' ISTOP=',ISTOP,' * MAX ITERATIONS *'
      ENDIF
      IF (ISTOP.EQ.3) THEN
        WRITE (1,'(A,I1,A)') ' ISTOP=',ISTOP,' * NO CONVERGENCE *'
      ENDIF

C     Change logarithmic model parameters to non-log parameters.

      DO 11 J=1,NP
        P(J)=EXP(P(J))
11    CONTINUE

C     Open output plot file

      OPEN (UNIT=2,FILE=PLTF)

C     Write the final chi-square sum and the final model into
C     the list file

      NL=(NP+1)/2
      WRITE (1,'(/)')
```

```
      WRITE (1,'(A,E10.4)') ' FINAL CHI-SQ. SUM IS CHI=',CHI
      WRITE (1,'(/)')
      WRITE (1,'(A)') ' THE FINAL MODEL PARAMETERS ARE:'
      WRITE (1,'(/)')
      WRITE (1,'(A,8(F8.2,A1))') ' rho:',(P(I),FIX(I),I=1,NL)
      WRITE (1,'(A,8(F8.2,A1))') '   d:',(P(I),FIX(I),I=NL+1,NP)
      WRITE (1,'(/)')

C     Write the data eigenvectors into the list file.

      WRITE (1,'(A)') ' DATA EIGENVECTORS:'
      WRITE (1,'(/)')

      DO 18 I=1,ND
        WRITE (1,'(I2,11F7.3)') I,(A(I,J),J=1,NF)
 18   CONTINUE

      WRITE (1,'(11I7)') (J,J=1,NF)
      WRITE (1,'(/)')

C     Write the parameter eigenvectors in the list file.

      WRITE (1,'(A)') ' PARAMETER EIGENVECTORS:'
      WRITE (1,'(/)')

      DO 19 I=1,NF
        WRITE (1,'(I2,11F7.3)') I,(V(I,J),J=1,NF)
 19   CONTINUE

      WRITE (1,'(11I7)') (J,J=1,NF)
      WRITE (1,'(/)')

C     Write the parameter eigenvalues in the list file.

      WRITE (1,'(A)') ' PARAMETER EIGENVALUES:'
      WRITE (1,'(/)')
      WRITE (1,'(11F7.3)') (W(I),I=1,NF)

C     Write the correlation matrix in the list file.

      WRITE (1,'(/)')
      WRITE (1,'(A)') ' CORRELATION MATRIX:'
      WRITE (1,'(/)')

      DO 20 J=1,NF
        DO 21 K=1,NF
          A(J,K)=0.
          DO 22 I=1,NF
            A(J,K)=A(J,K)+V(J,I)*V(K,I)/W(I)**2
 22       CONTINUE
 21     CONTINUE
        WRITE (1,'(I2,11F7.3)') J,(A(J,K)/SQRT(A(J,J)*A(K,K))
     &   ,K=1,J)
 20   CONTINUE

      WRITE (1,'(11I7)') (J,J=1,NF)
      WRITE (1,'(/)')

C     Write the AB/2 values, the measured and calculated apparent
C     resistivity values into the list file and the measured
C     apparent resistivity values into the plot file to be plotted
C     as small circles.

      WRITE (1,'(A,9X,A,7X,A,2(8X,A))') ' I','AB/2','Rhoam',
     & 'Rhoac','  WPM'
```

```
      DO 12 I=1,ND
        YM(I)=EXP(YM(I))
        YC(I)=EXP(YC(I))
        WRITE (1,'(I2,4F13.2)') I,X(I),YM(I),YC(I),1./SIG(I)
        XP=1.5*LOG10(X(I))+1.5
        YP=1.5*LOG10(YM(I))+1.0
        WRITE (2,'(A2,4F6.3,A4)') 'PS',XP,YP,0.1,0.0,' "!"'
12    CONTINUE

      CLOSE (UNIT=1)

C     Write the AB/2 and calculated apparent resistivity values into
C     the plot file to be connected by line segments.

      XP=1.5*LOG10(X(1))+1.5
      YP=1.5*LOG10(YC(1))+1.0
      WRITE (2,'(A2,2F6.3)') 'MA',XP,YP

      DO 13 I=2,ND
        XP=1.5*LOG10(X(I))+1.5
        YP=1.5*LOG10(YC(I))+1.0
        WRITE (2,'(A2,2F6.3)') 'PA',XP,YP
 13   CONTINUE

C     Write the model into the plot file to be plotted as a histogram.

      SP=0.0
      YP1=1.5*LOG10(P(1))+1.0
      WRITE (2,'(A2,2F6.3)') 'MA',1.5,YP1

      DO 14 J=1,NL-1
        SP=SP+P(NL+J)
        XP=1.5*LOG10(SP)+1.5
        WRITE (2,'(A2,2F6.3)') 'PA',XP,YP1
        YP=1.5*LOG10(P(J+1))+1.0
        WRITE (2,'(A2,2F6.3)') 'PA',XP,YP
        YP1=YP
  14  CONTINUE

      WRITE (2,'(A2,2F6.3)') 'PA',7.5,YP1
      WRITE (2,'(A4)') 'SP 1'
      WRITE (2,'(A)') 'SS "DEFAULT.SYM"'

C     Write the station identification, the model and the chi-square
C     sum into the plot file to be written on the plot.

      DO 15 I=1,12
        IF (PLTF(I:I).EQ.'.') GOTO 16
 15   CONTINUE

 16   IC=I-1
      STATION=PLTF(1:IC)
      XT=3.2
      YT=7.2
      HIGHT=0.22
      ANG=0.0
      WRITE (2,'(A2,4F6.3,A2,A8,A1)') 'PS',XT,YT,HIGHT,ANG,
     &  ' "',STATION,'"'
      XT=4.8
      YT=7.35
      HIGHT=0.15
      WRITE (2,'(A,4F6.3,A,I2,A)') 'PS',XT,YT,HIGHT,ANG,
     &  ' "',NL,' LAYERED MODEL"'
      XT=4.8
      YT=7.0
      HIGHT=0.125
```

```
      WRITE (2,'(A,4F6.3,A)') 'PS',XT,YT,HIGHT,ANG,
     &' "Layer   rho     d"'
      XT=5.0

      DO 17 J=1,NL-1
        YT=6.95-J*0.2
        WRITE (2,'(A2,4F6.3,A2,I1,1X,2F7.1,A1)') 'PS',
     &  XT,YT,HIGHT,ANG,' "',J,P(J),P(NL+J),'"'
17    CONTINUE

      YT=6.95-NL*0.2
      WRITE (2,'(A2,4F6.3,A2,I1,1X,F7.1,A1)') 'PS',
     &XT,YT,HIGHT,ANG,' "',NL,P(NL),'"'
      YT=6.95-(NL+1)*0.2
      XT=5.3
      HIGHT=0.11
      WRITE (2,'(A2,4F6.3,A,F5.3,A1)') 'PS',
     &XT,YT,HIGHT,ANG,' "Chisq =',CHI,'"'
      WRITE (2,'(A2,2F6.3)') 'SC',1.0,1.0
      WRITE (2,'(A2,2F6.3)') 'TR',0.0,0.0
      CLOSE (UNIT=2)

      RETURN
      END
```

```
C*************************************************************************
C                                                                        *
C                                SLUM                                    *
C                                                                        *
C       This program calculates apparent resistivity curve for a         *
C       given layered model rho(1),..,rho(NL),d(1),..,d(NL-1)            *
C       stored in the vector P. NL is the number of layers and must be   *
C       N<11. The apparent resistivity, stored in the vector Y, is       *
C       computed as a function of AB/2, stored in the vector X, by       *
C       using the gradient approximation and digital (J1) filter from    *
C       H.K. Johansen (1975). The resistivity transform, stored in T,    *
C       is convolved with the 141 point digital filter, stored in C.     *
C       The AB/2 values are equally distributed on log-scale with        *
C       10 points per decade.                                            *
C       The results are written into an output file and an output        *
C       plot file that can be plotted either on the terminal or as       *
C       a hard copy on a printer or a plotter.                           *
C                                                                        *
C*************************************************************************

        PROGRAM SLUM

        COMMON C(141)

        DIMENSION Y(41),X(41),T(181),P(19)
        CHARACTER*12 OUTF,PLTF

C       Read filenames

        WRITE (*,'(/)')
        WRITE (*,'(A,$)') '      OUTPUT FILE: '
        READ (*,'(A)') OUTF

        WRITE (*,'(A,$)') ' OUTPUT PLOT FILE: '
        READ (*,'(A)') PLTF

C       Call for the digital filter coefficients.

        CALL FLT

C       Read the model parameters from the terminal.

        WRITE (*,'(/)')
        WRITE (*,'(A,$)') ' NUMBER OF LAYERS:'
        READ (*,*) NL

        DO 11 I=1,NL-1
          WRITE (*,'(/)')
          WRITE (*,'(A,I1,A,$)') ' rho(',I,'):'
          READ (*,*) P(I)
          WRITE (*,'(A,I1,A,$)') '   d(',I,'):'
          READ (*,*) P(NL+I)
  11    CONTINUE

        WRITE (*,'(/)')
        WRITE (*,'(A,I1,A,$)') ' rho(',NL,'):'
        READ (*,*) P(NL)

C       Read minimum and maximum values of AB/2.

        WRITE (*,'(/)')
        WRITE (*,'(A,$)') ' TYPE (AB/2)min,(AB/2)max:'
        READ (*,*) XMIN,XMAX
```

```
C       Find the min and max values of I (ln(AB/2)=I*DX)
C       and the number of AB/2 values (NP+1) for which
C       RHOA is to be calculated.

        DX=0.2302585
        XMIN=ALOG(XMIN)/DX
        XMAX=ALOG(XMAX)/DX
        IMI=INT(XMIN)
        IMA=INT(XMAX)
        IF ((XMAX-REAL(IMA)).GE.0.5) THEN
          IMA=IMA+1
        ENDIF
        ND=IMA-IMI+1

C       Setting up the resistivity transform T(L).

        S=-1.7239458
        RX=EXP(DX)
        SL=EXP(S+DX*(IMA+101))
        RK=(P(NL-1)-P(NL))/(P(NL-1)+P(NL))

        DO 14 L=80,1,-1
          SLA=(RX**L)/SL
          AEXP=EXP(-2.*P(2*NL-1)*SLA)
          T(L)=P(NL-1)*(1.0-RK*AEXP)/(1.0+RK*AEXP)

          DO 13 J=NL-2,1,-1
            AEXP=EXP(-2.*P(NL+J)*SLA)
            WD=(1.0-AEXP)/(1.0+AEXP)
            T(L)=(P(J)*WD+T(L))/(1.0+WD*T(L)/P(J))
  13      CONTINUE

          IF (ABS(T(L)-P(NL)).LE.1.0E-02) THEN
            LMI=L
            GOTO 15
          ENDIF

  14    CONTINUE

  15    CONTINUE

        DO 17 L=81,140+ND
          SLA=(RX**L)/SL
          AEXP=EXP(-2.*P(2*NL-1)*SLA)
          T(L)=P(NL-1)*(1.0-RK*AEXP)/(1.0+RK*AEXP)

          DO 16 J=NL-2,1,-1
            AEXP=EXP(-2.*P(NL+J)*SLA)
            WD=(1.0-AEXP)/(1.0+AEXP)
            T(L)=(P(J)*WD+T(L))/(1.0+WD*T(L)/P(J))
  16      CONTINUE

          IF (ABS(T(L)-P(1)).LE.1.0E-02) THEN
            LMA=L
            GOTO 18
          ENDIF

  17    CONTINUE

  18    CONTINUE
        S1=0.0

        IF (LMI.GT.ND) THEN

          DO 19 L=1,LMI-ND
```

```
            S1=S1+C(L)
 19         CONTINUE

          ENDIF

          S2=0.0

          IF ((LMA+1-ND).LE.141) THEN

            DO 20 L=LMA+1-ND,141
              S2=S2+C(L)
 20         CONTINUE

          ENDIF

C         Convolve the resistivity transform T with the filter C.

          DO 22 I=1,ND
            Y(I)=0.0

            IF ((LMI-ND+I).GT.0) THEN
              S1=S1+C(LMI-ND+I)
            ENDIF

            L1=LMI+1+I-ND

            IF (L1.LT.1) THEN
              L1=1
            ENDIF

            L2=LMA-1+I-ND

            IF (L2.GT.141) THEN
              L2=141
            ENDIF

            DO 21 L=L1,L2
              Y(I)=Y(I)+T(L-I+ND)*C(L)
 21         CONTINUE

            Y(I)=Y(I)+P(NL)*S1+P(1)*S2

            IF ((LMA+I-ND).LE.141) THEN
              S2=S2-C(LMA+I-ND)
            ENDIF

 22       CONTINUE

          DO 23 I=1,ND
            X(I)=RX**(IMI-1+I)
 23       CONTINUE

C         Write the results in output files.

          CALL SLWROT (X,Y,P,ND,NL,OUTF,PLTF)

          STOP
          END
```

```
      SUBROUTINE SLWROT(X,Y,P,ND,NL,OUTF,PLTF)

*************************************************************************
*                                                                       *
*                              SLWROT                                   *
*                                                                       *
*     This routine writes out the results from the forward program      *
*     SLUM into the output file OUTF and a plot file PLTF that          *
*     can be plotted on the screen or as a hard copy on a printer       *
*     or a plotter.                                                     *
*                                                                       *
*************************************************************************

      DIMENSION X(41),Y(41),P(19)

      CHARACTER*12 OUTF,PLTF

C     Open output file

      OPEN (UNIT=1,FILE=OUTF)

C     Write the AB/2 and the calculated apparent resistivity
C     values, RHOA, in the output file.

      DO 11 I=1,ND
        WRITE (1,'(3F10.2)') X(I),Y(I)
 11   CONTINUE

      CLOSE (UNIT=1)

C     Open output plot file

      OPEN (UNIT=2,FILE=PLTF)

C     Write the AB/2 and apparent resistivity values in the plot file
C     to be plotted as small circles.

      DO 12 I=1,ND
        XP=1.5*LOG10(X(I))+1.5
        YP=1.5*LOG10(Y(I))+1.0
        WRITE (2,'(A2,4F6.3,A4)') 'PS',XP,YP,0.1,0.0,' "!"'
 12   CONTINUE

C     Write the AB/2 and apparent resistivity values in the plot file
C     to be connected by line segments.

      XP=1.5*LOG10(X(1))+1.5
      YP=1.5*LOG10(Y(1))+1.0
      WRITE (2,'(A2,2F6.3)') 'MA',XP,YP

      DO 13 I=2,ND
        XP=1.5*LOG10(X(I))+1.5
        YP=1.5*LOG10(Y(I))+1.0
        WRITE (2,'(A2,2F6.3)') 'PA',XP,YP
 13   CONTINUE

C     Write the model into the plot file to be plotted as a histogram.

      SP=0.0
      YP1=1.5*LOG10(P(1))+1.0
      WRITE (2,'(A2,2F6.3)') 'MA',1.5,YP1

      DO 14 J=1,NL-1
        SP=SP+P(NL+J)
```

```
          XP=1.5*LOG10(SP)+1.5
          WRITE (2,'(A2,2F6.3)') 'PA',XP,YP1
          YP=1.5*LOG10(P(J+1))+1.0
          WRITE (2,'(A2,2F6.3)') 'PA',XP,YP
          YP1=YP
 14     CONTINUE

        WRITE (2,'(A2,2F6.3)') 'PA',7.5,YP1
        WRITE (2,'(A4)') 'SP 1'

C       Write the model into the plot file to be displayed numerically.

        WRITE (2,'(A)') 'SS "DEFAULT.SYM"'
        XT=4.8
        YT=7.35
        HIGHT=0.15
        ANG=0.0
        WRITE (2,'(A,4F6.3,A,I2,A)') 'PS',XT,YT,HIGHT,ANG,
     &  ' "',NL,' LAYERED MODEL"'
        XT=4.8
        YT=7.0
        HIGHT=0.125
        WRITE (2,'(A,4F6.3,A)') 'PS',XT,YT,HIGHT,ANG,
     &  ' "Layer   rho     d"'
        XT=5.0

        DO 15 J=1,NL-1
          YT=6.95-J*0.2
          WRITE (2,'(A2,4F6.3,A2,I1,1X,2F7.1,A1)') 'PS',
     &    XT,YT,HIGHT,ANG,' "',J,P(J),P(NL+J),'"'
 15     CONTINUE

        YT=6.95-NL*0.2
        WRITE (2,'(A2,4F6.3,A2,I1,1X,F7.1,A1)') 'PS',
     &  XT,YT,HIGHT,ANG,' "',NL,P(NL),'"'
        WRITE (2,'(A2,2F6.3)') 'SC',1.0,1.0
        WRITE (2,'(A2,2F6.3)') 'TR',0.0,0.0
        CLOSE (UNIT=2)

        RETURN
        END
```