U.S. DEPARTMENT OF THE INTERIOR
GEOLOGICAL SURVEY

Nonlinear least-squares inversion of infinite line source data (Program NLSINF)

by

Walter L. Anderson *

Open-File Report 89-555

1989

DISCLAIMER

This program was written in Fortran-77 for an HP-9000 (series-370) computer using the HP-UX (Unix) operating system +. Although program tests have been made, no guarantee (expressed or implied) is made by the author regarding program correctness, accuracy, or proper execution on all computer systems.

+ Any use of trade names in this report is for descriptive purposes only and does not imply endorsement by the U.S. Geological Survey. This report is preliminary and has not been reviewed for conformity with U.S. Geological Survey editorial standards.

* Denver, Colorado.

CONTENTS

ABSTRACT

A computer program is presented that inverts electromagnetic sounding data for an infinite line source on or above a layered earth using an adaptive nonlinear least-squares method. Options are provided to use separate or multiple field components (Hx, Hz) or polarization ellipse data (tilt angle, ellipticity) in the inversion algorithm. Joint inversion of frequency and/or geometric sounding data is performed efficiently using a fast lagged convolution integration routine. An inversion example for a known model is discussed, where the input data was generated from a corresponding stand-alone forward program. Program parameters are defined, and the Unix operating instructions are summarized. The Fortran source code is available on magnetic tape upon request.

INTRODUCTION

A computer program for the inversion of infinite line source sounding data over assumed layered earth models is discussed in this report. The numerical technique is based on a general adaptive nonlinear least-squares algorithm originally developed by Dennis and others (1979; 1981), and extended externally for constrained nonlinear regression by Anderson (1982). Solution of the forward problem is described mathematically in terms of sine and cosine transforms by Ward and Hohmann (1987; pp.242-246) for a line source of current on or above a layered earth. The forward problem, which is required iteratively for the inverse solution, is calculated using an efficient numerical digital filtering algorithm (called lagged convolution) by Anderson (1975). Note that a stand-alone forward program (FWDINF) can be easily obtained directly from the inversion program (NLSINF), and is described in general by Anderson (1984) for converting any inversion program to a forward program, or vise-versa. (See Appendix 1 for a brief discussion on usage of the associated forward program FWDINF.)

In this report, I utilize the general nonlinear least-squares (NLS) method defined by Anderson (1982), but only as it applies here to infinite line source sounding data. The infinite wire source is assumed oriented along the y-axis in a right-handed Cartesian coordinate system with +z pointing downward in the earth and z=0 at the surface. Various options are provided for using separate or multiple field components (Hx, Hz) or polarization ellipse data (tilt angle and ellipticity) as joint inversions in the NLS solution. In practice, either amplitude, phase, real, or imaginary parts of Hx and/or Hz can be used. An option that combines ratios of Hx and Hz is provided by using tilt angle and ellipticity as described in Smith and Ward (1974). Either parametric (varying frequency) or geometric (varying x-distance) sounding data may be selected using a generalized data matrix input scheme.

The remainder of this report contains 1) a summary of the general computations, 2) a description of the required program parameters, and 3) the Unix operating instructions. Appendix 1 contains a discussion on the usage of the stand-alone forward program (FWDINF), including an example problem and associated plot; Appendix 2 contains some suggestions for converting the programs to other computer systems; Appendix 3 contains a listing of an input/output inversion example; and Appendix 4 contains information on obtaining the Fortran source codes on magnetic tape.

SUMMARY OF CALCULATIONS

The formulas for evaluating the magnetic field components Hx or Hz for an infinite line source of current parallel to the y-axis and located at (0,0,-h) over a layered earth are derived by Hohmann and Ward (1987; pp. 242-246) in terms of sine and cosine transform integrals. (The Ey field is also given, but has been excluded in this version of program NLSINF.) For any receiver point (x,0,-z), |x|>0, the magnetic fields are given in induction-number argument form by

$$H_x(B) = -I/c \;\langle\mathrm{Int}\rangle\; [(1+r_{TE})\exp(-u_0|z|)] \cos(gB)\, dg \qquad (1)$$

and

$$H_z(B) = -I/c \;\langle\mathrm{Int}\rangle\; [(1+r_{TE})\exp(-u_0|z|)] \sin(gB)\, dg \qquad (2)$$

where

induction no. $B = x/\delta_j$, skin depth $\delta_j = (2/\sigma_j \mu_0 w)^{1/2}$

σ_j = conductivity (S/m) of layer j with

j = 1 if h = 0 (wire on surface), or
j = 2 if h < 0 (wire above surface),

$c = 2(pi)\delta_j$, $u_0 = g/\delta_j$, $\mu_0 = 4(pi)10^{-7}$, $w = 2(pi)f$,

I = current (A), f = frequency (Hz.), (pi)=3.14159...

<Int> = integral from 0 to infinity, and

r_{TE} = reflection coefficient for TE mode (eq. 4.47 in Ward and Hohmann, 1987), defined as a recurrence relation in terms of layer thicknesses and conductivitites.

If the wire source is actually suspended above the surface (e.g., placed on telephone poles), then this configuration can be selected by using a first layer with near-zero conductivity and thickness |-h| to simulate an air layer; in this case, j=2 denotes the first earth layer, which avoids possible division by 0 in the skin depth used in equations (1) and (2). The effect of wire source elevation is usually minimal for most models using moderate to large frequencies and x-spacings (intermediate to far-field).

Equations (1) and/or (2) are evaluated using a fast lagged convolution digital filtering algorithm by Anderson (1975), and becomes especially efficient when only geometric (x-varying) sounding data are used. In this case, the data matrix is scanned for the entire x-range, and only one evaluation of equations (1) or (2) is needed in the lagged convolution; subsequent values within the x-range needed in the forward solution are simply spline-interpolated as required. A similar procedure is used for evaluation of partial derivatives of equations (1) and/or (2) needed in the NLSINF inversion program.

The NLS method (Anderson, 1982; Dennis and others, 1979, 1981) requires a twice-continuously differentiable nonlinear objective function describing the model equation as a function of the unknown parameters. For this program, the objective function is expressed in terms of operations (e.g., tilt and ellipticity) using equations (1) and/or (2), where the model parameters are the conductivities and thicknesses of an arbitrary number of layers (maximum number of layers is set to 10 in this program).

The unknown layered model parameters used in NLSINF are denoted by the vector b(j), j=1,2,...,k, where k=2*mm, mm=number of layers. The order of parameters to be estimated initially in b(j) are given in Table 1.

Table 1.-- Definition of model parameters

layer	conductivities (S/m)	thicknesses (m)
1	b(1) = σ_1	b(mm+1) = h_1
2	b(2) = σ_2	b(mm+2) = h_2
...	...	...
mm-1	b(mm-1) = σ_{mm-1}	b(2*mm-1) = h_{mm-1}
mm	b(mm)= σ_{mm}	* b(2*mm) = shift

* k = 2*mm is the total number of parameters.

Note that in Table 1, there are 2*mm-1 layer parameters and a scale or shift parameter b(2*mm), which is used to change signs and/or scale the calculated soundings to match the observed data. Generally, b(2*mm)=-1 or +1 and should be fixed in the NLS program. Another use for the shift is to normalize a frequency sounding at a fixed x-distance with the constant primary field H0=-I/2(pi)x when mutual coupling ratios are observed; in this case, the shift should be set to 1/H0. In some cases, the shift could be an unknown parameter to be determined by the NLS program. If mm=1, indicating a homogeneous half-space model, then the layer thicknesses are absent in the b(k) vector.

Many NLS options are available in the interface subprogram NLSOL (Anderson, 1982, p. 11-21), which the reader should become familiar with before attempting to run NLSINF. Program NLSINF calls a Unix version of VAX subprogram NLSOL (called NLSOL2), and is identical in design and operation to that described by Anderson (1982).

Following the NLS notation in Anderson (1982, p.11-12), we let x(i,1) be the observed independent variable (described later) and y(i) be the corresponding dependent variable for the i-th observation, where each data array x(i,1), y(i) is given for i=1,2,...,n, n>k. In general, a data set should be given in ascending x(i,1) order for all i, which is useful when plotting the observed data and calculated least-squares fit. However, concatenated data sets, each with different but increasing x(i,1), can be used in the observed data matrix. Other independent variables in a data matrix row define specific options and are given in arrays x(i,2), x(i,3), and x(i,4). The various data matrix options available for program NLSINF are completely defined below in the section "DATA MATRIX OPTIONS".

Program NLSINF reads the observed data matrix in n rows in the following order:

(y(i),(x(i,j),j=1,4),i=1,n)

using an arbitrary object or run-time input format (see any Fortran manual).

Because the dependent variable y(i) can be in different units and ranges, it is advised that a weighted least-squares option be used (see iwt=1 or 2, Anderson, 1982, p.14-15), which requires the augmented data matrix

(y(i),(x(i,j),j=1,5),i=1,n)

where x(i,5) is the standard deviation (iwt=1) of observation y(i), or x(i,5) is the variance (iwt=2). Note that if x(i,5) is unknown, one may use the statistical weighting factor 1/y(i) (Bevington, 1969, p.108) by setting x(i,5)=y(i) and iwt=2; this procedure is preferable to using unity weights (iwt=0).

An analytical partial derivative subprogram (pcode; see Anderson, 1982) is not included in this version of NLSINF. Therefore the estimated finite-difference derivative option (ider=1) must be used, which requires only the forward problem solution subprogram (fcode).

Because realizable layered earth models are sought to fit the given data, a constrained minimization type (sp=3 or 4) is advised, along with an initial guess array b(j) and reasonable lower and higher bound arrays, bl(j) and bh(j) respectively, where bl(j)<=b(j)<=bh(j), j=1,2,...,k (see Anderson, 1982, p.17). This approach limits parameter space searching, and in some cases may avoid false starts or catastrophic overflow conditions from poor initial estimates and/or noisy data. In addition, individual parameters can be held fixed in the least-squares program by specifying parameters ip and ib (Anderson, 1982, p.13). For example, this should always be done if a parameter is known in advance, such as the shift parameter, b(2*mm); also, in some cases, it is helpful to fix a parameter that cannot be adequately resolved with the given data matrix.

PARAMETERS, FILES AND DATA REQUIRED

Two general classes of namelist parameters are required: $parms and $init. All the $parms parameters (excluding the istop=0 option), program files (Fortran units 5 to 16) and data ordering requirements used by NLSINF are identical to those described in detail for VAX subprogram NLSOL (Anderson, 1982, p.9-21). (Familiarity with all the $parms defined in the latter reference is assumed; definitions of these parameters are not repeated here in the interest of brevity.) Note, however, that the ordering of the $parms estimated parameter vector b(j) used by NLSINF must be given exactly as described in Table 1. The $init model parameters required by NLSINF (defined below) must be given immediately after the run-time format statement in file 5 (see Anderson, 1982, p.10, item 5). A Unix script is used to equate units 5 to 16 to arbitrary file names in a subdirectory, and is further described in the UNIX OPERATING INSTRUCTIONS section. For some typical input data sets, refer to the EXAMPLES section and to Appendix 3.

$init PARAMETER DEFINITIONS

$init parameters:

mm ... is the number of layers as defined in Table 1. An arbitrary range of mm is set at 0<mm<11 (default mm=1). When selecting this $init option, it is required that $parms parameter k=2*mm is also explicitly given. (This is a dual requirement that is not checked, however it is necessary because a general purpose nonlinear least-squares routine NLSOL2 is being used to interface with a layered model problem.)

z ... is the z-coordinate at any receiver point (x,0,z), and must be given such that z<=0.0 (default z=0.0). See $init parameter iair=1 (below) for possible problems with specifying the z elevation coordinate.

tol ... is the lagged convolution integration tolerance in Anderson (1975); default tol = 1E-9, which is about the best accuracy possible for a 32-bit system. To run somewhat faster, tol can be increased (e.g., tol=1E-6).

iair ... is an air simulation option for layer 1. If iair=0 (default), then the infinite wire source is assumed to be on the earth′s surface; this also implies that the definition of skin depth following equations (1) and (2) uses the conductivity of layer 1. In this case z<=0.0 and mm>=1 are allowed.

If iair=1 is selected, then the first layer (assumed air) must be set with a very small conductivity (e.g., fix b(1)=1e-15), and the second layer conductivity b(2) is used to calculate the skin depth in equations (1) and (2); in this case, iair=1 also implies that mm>1, but z is now the elevation on or above the "fictitious air layer" of thickness b(mm+1). Thus the iair=1 option puts the effective receiver elevation at -[|z|+b(mm+1)], where b(mm+1) is the (fixed) air layer thickness, and z<=0.0. An error message is given if mm=1 and iair=1 are selected; however, an error in the correct z value cannot be noted.

iall ... is a special option to produce an output file containing all components of a computed forward curve in one pass. If iall=0 (default), then this option is ignored. If iall=1, then for each forward curve, a list-directed (formatted) file will contain: total observation count, a heading (maximum 40 characters), followed by the field component amplitude, phase, real, and imaginary components (or tilt angle and ellipticity) corresponding to each frequency or x-distance selected; these records are then followed by another record containing x(i,j),j=1,4 for the i-th observation in the data matrix row (see DATA MATRIX OPTIONS below). The iall=1 option is only intended for subsequent use (e.g., plotting) following the forward program (FWDINF), and is NEVER recommended when using the inversion program (NLSINF).

$end [end of $init parameters; the "end" is optional.]

DATA MATRIX OPTIONS

The data matrix (discussed following Table 1) is read under the run-time format statement, and is defined as the sequence of ordered rows:

(y(i),(x(i,j),j=1,m*),i=1,n),

where m*=m=4 if iwt=0 (default), or m*=m+1 if iwt=1 or 2. Note that the number of independent variables is always m=4 for program NLSINF. The data matrix is read on logical unit ialt (default 10) using a run-time format statement (see any Fortran manual). The number of items read per record depends on $parms m and iwt. The various data matrix options are summarized as follows:

(a) "Frequency soundings" are defined when x(i,1)>0.0 and x(i,4)>0.0; in this case, x(i,1) is the i-th frequency (in Hertz) and x(i,4)=+r, where r=x is the orthogonal distance from the infinite wire source oriented along the y-axis. Case (a) requires $parms m=4 with the following items per record:

y(i)= i-th observed dependent variable as defined by x(i,2) and x(i,3).

x(i,1)= i-th observed frequency (Hertz), where x(i,1)>0.0 and should be given in ascending order for a fixed distance r=x(i,4). Note that stacked frequency soundings can be given for separate (fixed) distances simply by appending each curve in the data matrix.

x(i,2)= defines the field component type; set x(i,2)=1.0 for Hx, =2.0 for Hz, or =3.0 if both Hx and Hz are used, e.g., if tilt angle and ellipticity are given in y(i).

x(i,3)= defines the y(i) data type; set x(i,3)=1.0 for amplitude (unnormalized), =2.0 for phase (in degrees), =3.0 for real-part, =4.0 for imaginary-part, =6.0 for tilt angle (in degrees), or =7.0 for ellipticity. Note that the value 5.0 is currently not used in program NLSINF, and is reserved for possible future use.

x(i,4)= i-th observed distance r=x(i,4)>0.0 (m) from the infinite wire source and should be constant for all x(i,1) within a given frequency sounding.

x(i,5)= weight factor of i-th observation (include only if $parms iwt>0).

(b) "Geometric soundings" are defined when x(i,1)>0.0 and x(i,4)<0.0; in this case, x(i,1) is the i-th r distance (in m) and x(i,4)=-f, where f is the frequency (in

Hertz). Note that the sign of x(i,4) is the key in selecting case (a) or (b). It is very important that the user understand this distinction. Case (b) requires $parms m=4 with the following items per record:

y(i)= i-th observed dependent variable as defined by x(i,2) and x(i,3).

x(i,1)= i-th observed distance (m), where x(i,1)>0.0 and should be given in ascending order for a fixed frequency x(i,4)=-f. Note that stacked distance soundings can be given for separate (fixed) frequencies simply by appending each curve in the data matrix.

x(i,2)= defines the field component type; set x(i,2)=1.0 for Hx, =2.0 for Hz, or =3.0 if both Hx and Hz are used, e.g., if tilt angle and ellipticity are given in y(i).

x(i,3)= defines the y(i) data type; set x(i,3)=1.0 for amplitude (unnormalized), =2.0 for phase (in degrees), =3.0 for real-part, =4.0 for imaginary-part, =6.0 for tilt angle (in degrees), or =7.0 for ellipticity. Note that the value 5.0 is currently not used in program NLSINF, and is reserved for possible future use.

x(i,4)= i-th observed frequency as x(i,4)=-f<0.0 (Hertz) from the infinite wire source and should be constant for all x(i,1) within a given geometric sounding.

x(i,5)= weight factor of i-th observation (include only if $parms iwt>0).

For a given data set, (a) or (b) above, and for each separate sounding within the data matrix, the observations should be ordered by increasing x(i,1) for i=1,2,...,n. This also aids in plotting the results after running NLSINF. In case (b), when both tilt angle and ellipticity data are given, it is generally convenient to append each uniform data type, x(i,3)=6 or 7, and to keep x(i,1) increasing within each data matrix subset. However, any arbitrary order can be used if desired, since each row in the data matrix is essentially independent of all others.

As suggested earlier, it is recommended that the weighted observations option iwt>0 (Anderson, 1982, p. 14-15) be used with this program; usually iwt=2 is adequate for most cases. (See EXAMPLES in the next section, and in Appendix 3.)

EXAMPLES OF INPUT PARAMETERS AND DATA ORDERING

(In this section I assume that the reader is familiar with ALL the $parms definitions as given by Anderson, 1982, p.11-19.)

1. Normalized Hz/H0 frequency sounding, weighted observations (iwt=2), and alternate input data file (ialt=5) for reading the data matrix along with the input parameters on unit 5:

```
example 1.  amplitude and phase data.
$parms n=42,m=4,sp=3,iprt=-1,
k= 2,ip= 1,ib= 2, ialt=5,
ider=1,iwt=2,
niter= 40,
bl=1*.1e-7, -1e5,
bh=1*10, 1e5,
b= .1, -6283.183061$
(5g10.5,t1,g10.5)
0.9962        .25e-2     2.000     1.000     1000.
0.777         .2533      2.000     1.000     1000.
<etc. for 19 more obs. where y(i)=amplitude and x(i,3)=1.0>
-.811         .25e-2     2.000     2.000     1000.
-4.9          .2533      2.000     2.000     1000.
<etc. for 19 more obs. where y(i)=phase and x(i,3)=2.0>
$init mm= 1,z= 0,iair= 0$
```

Note: Since iwt=2 and m=4, 6-columns (1-dependent, 4-independent, 1-weight) are required in the data matrix row, where in this case, the last item read represents a weight function using y(i) to compute a weight wt(i)=1/y(i). Observe that the object format uses "t1" to position the next format back at column 1 and to reread y(i) into x(i,5), which is replaced by 1/x(i,5) by subprogram NLSOL2 since iwt=2 (see Anderson, 1982, p. 14-15).

2. Geometric sounding at 60 Hertz, weighted observations (iwt=2), air layer case, and alternate input data file (ialt=5) for reading the data matrix along with the input parameters on unit 5:

```
example 2.  tilt angle and ellipticity data.
$parms n=22,m=4,sp=3,iprt=-1, ialt=5,
k= 4,ip= 3,ib= 1,3,4,
ider=1,iwt=2,
niter= 40,
bl=1e-15,.1e-7, 1, -1e5,
bh=2*10, 1000, 1e5,
b= 1e-15,.1,12, -1$
(5g10.5,t1,g10.5)
58.309       100.00     3.000     6.000     -60.0
41.61        158.45     3.000     6.000     -60.0
<etc. for 9 more obs. where y(i)=tilt angle and x(i,3)=6.0>
-.27         100.00     3.000     7.000     -60.0
-.301        158.45     3.000     7.000     -60.0
<etc. for 9 more obs. where y(i)=ellipticity and x(i,3)=7.0>
$init mm= 2,z= -10,iair= 1$
```

(The note under example 1 applies to example 2 as well.)

SPECIAL OBJECT FORMAT PHRASES

If an existing data matrix file does not have the proper defined column ordering of the form y(i),(x(i,j),j=1,m), then the Fortran "tn" format phrase (as used in the above examples) may be used to begin at any column n in the data record. For example, the format (t61,f10.0,t1,5f10.0) will select y(i) using column 61-70 and x(i,1) beginning at column 1. See any Fortran-77 coding manual for other allowable object (run) time format phrases (e.g., the g-format, use of "/" to skip records, etc.). Note that "tab"-characters must NOT be used when creating the data matrix file.

UNIX OPERATING INSTRUCTIONS

The VAX/VMS instructions described by Anderson (1982, p.22-24) cannot be followed to run NLSINF on a Unix system. The Unix environment is significantly different from VMS, and a new approach has been developed to run NLSINF using a general csh (c-shell) script, called run.com. The contents of the script and notes on how to run NLSINF (or any nls- or fwd-program) are as follows:

```
############################################################
# run.com -- general script usage: run $1 $2 -- interpreted:
# run [pgm] [id] -- to run program (pgm) with passed parm=id.
# assumes alias run 'csh -f run.com !* &' or equivalent.
#           ----------------------------
# note run.com executes: cd ~/p/$1 and ~/lib/$1 $2; but
# any other directories can be used in run.com as desired.
# to use run.com from any working directory, simply remove
# the cd ~/p/$1 line below, then all input/output files will
# be stored in the current working directory.
#
```

```
# e.g.,  first prepare input files
#        ~/p/pgm/test.par and ~/p/pgm/test.dat, etc.
# then:  run pgm test -- output files will be
#        ------------
#        ~/p/pgm/test.out and ~/p/pgm/test.log, etc.
###########################################################
onintr exit
cd ~/p/$1
date >! $2.log; time >>! $2.log; pwd >>! $2.log
~/lib/$1 $2 |& tee >>! $2.log
if ($status != 0) then
  echo "error exit in $1" |& tee >>! $2.log
  if (-e core*) rm core*
  goto exit
endif
if (-e for005.tmp && -e $2.par) mv -f for005.tmp $2.par
if (-z $2.plt) rm -f $2.plt
if (-z $2.all) rm -f $2.all
  exit:
pwd >>! $2.log; time >>! $2.log; date >>! $2.log
###########################################################
```

The above script makes several assumptions concerning where files are stored. The comments at the head of run.com basically gives all the pertinent information for making changes and preparing input files. Note that the user's home directory (denoted by ~) can be any local path, and the executable program (nlsinf in this case) is assumed to be in the directory ~/lib/nlsinf, whereas input and output files are stored in the directory ~/p/nlsinf; this means any data set name (or id) can be used to uniquely define runs, e.g., ~/p/nlsinf/mydata.par can be the required parameter file, and ~/p/nlsinf/mydata.dat the corresponding data matrix file (if ialt is not 5). The ~/p subdirectory is used as a switching point for other programs or processes (thus the name "p").

After preparing the above input files, then program NLSINF can be run simply by invoking the Unix script as:

```
run nlsinf mydata
```

This assumes that the user has setup the "run" alias as noted in the run.com comments. The resulting output files ~/p/nlsinf/mydata.out and ~/p/nlsinf/mydata.log will be stored, unless severe errors in the parameter or data files exist. Errors are generally self-explanatory and noted in the .log file.

If the program runs normally to completion, then a temporary restart parameter file (called for005.tmp) is created, which contains the last solution b-vector appended to the original $parms namelist in the previous .par file. This is a convenient tool to rerun NLSINF, particularly if a convergent solution has not been achieved (e.g., niter was reached). The run.com script, as presently coded, automatically renames file for005.tmp to the original parameter file name. Thus to continue more iterations (or make any other parameter changes), simply invoke the script again. It should be noted that $parms istop=0 (Anderson, 1982, p.14) cannot be used because the .par file (unit=5) is positioned at the end-of-file when creating for005.tmp.

Notice that the run.com script is set to execute in background mode (see the "&" at the end of the run alias). This script was purposely established to be non-interactive (i.e., no online prompts and responses), and is helpful if the user wishes to do other tasks while the background job is running. However, to run the script online, simply redefine the alias without the ending "&". If a severe error occurs such that the Unix $status bit is set, and a core dump is given, then the run.com script will delete any (useless?) core dumps.

The main output file has a suffix .out, and is similar to the .log output, but without certain system messages (also see $parms iprt in Anderson, 1982, p.15). The "time" lines in the .log file give statistics about the run-time for the given program. The .out file may be printed or used as an input file to a plot routine; e.g., to plot the observed (obs.), calculated (cal), and residual (res) curves.

ERROR MESSAGES

Almost all $parms syntactical errors are flagged and printed on the .log and .out files, and the job is aborted (see Anderson, 1982, p.24 for other comments). However, some cross references (or dual inputs) are not checked; for example, the relationship between $parms k (k=2*mm) and $init mm is not double checked by program NLSINF. This is because a general-purpose nonlinear least-squares algorithm (NLSOL) is being used as a control program, but the $init model input is external to the particular nonlinear problem requirements (NLSINF) read by subprogram SUBZ (see Anderson, 1982, p.38). Therefore, the user is responsible for providing exactly k parameter estimates in b(i),k=1,2,...,k (see Table 1), and that k=2*mm, otherwise unpredictable results could occur that are unchecked.

PRINTED OUTPUT

All input parameters are written on files .out and .log with the $init parameters given first, followed by all $parms parameters given or assumed by default. (Refer to Appendix 3 for a complete sample output listing.)

Specific names (e.g., fmt, it, nf, ...) used by NLSOL in the output listings are tabulated in Anderson (1982, p.25-26). Program NLSINF also provides a summary listing of the final solution vector b() and parameter names at the end of the output file.

REFERENCES

Anderson, W.L., 1975, Improved digital filters for evaluating Fourier and Hankel transform integrals: Nat. Tech. Info. Service (NTIS) Report PB-242-800, 223 p.

---------, 1982, Adaptive nonlinear least-squares solution for constrained or unconstrained minimization problems (Subprogram NLSOL): USGS Open-File Rept. 82-68, 65 p.

---------, 1984, A general interface for producing forward solution programs (Subprogram FWDSOL): USGS Open-File Rept. 84-348, 43 p.

Bevington, P.R., 1969, Data reduction and error analysis for the physical sciences: McGraw-Hill, N.Y., 336 p.

Dennis, J.E., Gay, D.M., and Welsch R.E., 1979, An adaptive nonlinear least-squares algorithm: Univ. of Wisconsin MRC Tech. Sum. Rept. 2010 (also available as NTIS Rept. AD-A079-716),

---------, 1981, An adaptive nonlinear least-squares algorithm: ACM Trans. on Math. Software, v.7, no.3, p.348-368.

Smith, B.D., and Ward, S.H., 1974, On the computation of polarization parameters: Geophysics, v.39, no.6, p.867-869.

Ward, S.H., and Hohmann, G.W., 1987, Electromagnetic theory for geophysical applications, in: Nabighian, M.N., Ed., Electromagnetic methods in applied geophysics, v.1, Soc. Expl. Geophysics.

Appendix 1.-- Forward program FWDINF usage

As mentioned in the introduction, a stand-alone forward program (FWDINF) corresponding to the inverse program (NLSINF) is also available, and is always distributed along with NLSINF as discussed in Appendix 4. The development of an associated forward program based on the same subroutines as used in the inverse program is described by Anderson (1984). It is obvious that the inverse solution must compute repeated forward solutions in the least-squares algorithm. Thus repackaging part of the inverse solution as a separate but compatible forward program is almost trivial.

Following Anderson (1984, p.5), the forward program (FWDINF) is produced from the inverse program simply by replacing subprogram NLSOL2 with subprogram FWDSOL, along with a few other minor modifications to the main program (e.g., Unix file handling). The subprograms FCODE and SUBZ used in NLSINF and FWFINF are now identical, and thus the $init model parameters described in this report apply to both programs. However, the $parms used by NLSOL (Anderson, 1982) are not used in FWDINF, but are replaced by $fwd parameters as defined by Anderson (1984, p.7-8).

A summary of the $fwd parameters is repeated here, but more information can be found in Anderson (1984). The concepts described above in the DATA MATRIX OPTIONS section should be kept in mind to understand how and why certain parameters are required in the $fwd namelist.

$fwd parameters:

mm ... is the number of layers in the model (if mode=1 or -1 defined below), or the number of parameters given in array b()--used only if mode=0. Note that $fwd mm should ALWAYS be chosen the same as any $init mm definition, and is a dual requirement that cannot be cross-checked by FWDSOL.

mode ... =1 (default) implies a layered model as given in arrays sig() and h(); mode=-1 implies a layered model as given in arrays rho() and h(); mode=0 implies exactly mm-values (no layers) are given in array b(). NOTE: mode=1 must be used for all EM-programs; mode=-1 must be used for DC-programs and MT-programs; mode=0 is a special case for other uses not using layer arrays sig or rho. The mode option is simply used for a fwd-program to pack array b() with either sig() and h(), or rho() and h(), as consecutive parameters as defined in the corresponding NLS-inversion program. (Because FWDINF is an EM program, mode=1 is ALWAYS required here.)

sig() ...conductivity (S/m) array for mm layers when mode=1.

rho() ...resistivity (ohm-m) array for mm layers when mode=-1. A mapping to the opposite mode will be given in the .out print file as additional information, but this does not imply that the proper sig() or rho() is switched if mode is improperly set -- see note above about mode requirements.

h() ... thickness (m) array for mm-1 layers when mode=1 or -1; if mm=1 (halfspace), then array h() is not required.

shift ...multiplier "shift" factor defined by some programs to scale the computed response or amplitude function. Generally, the shift is to be stored in array element b(2*mm); this is done automatically by the FWD-program interface. If a "2nd shift" is needed in b(2*mm+1), then this b() element must be given in $fwd separately.

b() ... input array passed to subprograms fcode and subz (same as used in any NLS-inversion program). This array is automatically obtained from sig(), rho(), and h() when mode=1 or -1; however, when mode=0, then b() must be given explicitly in the $fwd input.

x1 ... initial x(i,1) for i=1 in the generated data matrix. Refer to the corresponding NLS-inversion program for the x1 type; e.g., for FWDINF, x1 is either frequency or distance as defined above in the DATA MATRIX OPTIONS section. However, note that x1 is required only when nx>0 (defined below).

nx ... is the number of points per decade to generate from x1 to xm. Use nx>0 to select the x1-to-xm option. Use nx<0 to select specific x(i,1)>0 ascending points in array xnx().

xm ... final x(i,1) for i=n (n determined automatically) in the generated data matrix. Make sure xm>=x1 when nx>0 is selected.

xnx() .. array of specific x(i,1) points to use for x1 in the generated data matrix for i=1,|nx|. Observe that xnx(i) is required in ascending order only when the nx<0 option is selected.

x2 ... (default 1.0) is the x(i,2) constant to use in the generated data matrix for all i=1,n. e.g., for FWDINF, x2 is the component type (x2=1 for Hx, x2=2 for Hz, or x2=3 for Hx and Hz as required for tilt angle and ellipticity option). Again, refer to the DATA MATRIX OPTIONS on use of x2=x(i,2) for cases (a) or (b).

x3 ... (default 1.0) is the x(i,3) constant to use in the generated data matrix for all i=1,n. Refer to the

DATA MATRIX OPTIONS on use of x3=x(i,3) for cases (a) or (b).

x4 ... (default 1.0) is the x(i,4) constant to use in the generated data matrix for all i=1,n. Refer to the DATA MATRIX OPTIONS on use of x4=x(i,4) for cases (a) or (b). Since the sign of x4 and its value is a key option on the meaning of x1 or xnx(), the default x4 should always be overridden.

iplt ... =0 (default) to NOT create an output plot file with suffix .plt; however, an output file with suffix .out is always generated containing the computed curve with x(i) the independent variable via x1-to-xm or xnx() parameter, and y(i) is the computed dependent variable via parameters selected in x2, x3, and x4. Use iplt=1 to create plot data (Anderson, 1984, p.16-17) on file .plt with the computed x(i)=x(i,1) and y(i) arrays as defined above.

xt ... 'x' (default); use up to 40-characters for any xaxis plot title.

yt ... 'y' (default); use up to 40-characters for any yaxis plot title.

$end [end of $fwd parameters; the "end" is optional.]

The procedure to specify any number of stacked forward soundings is accomplished simply by preparing sets of namelist lines in an input parameter file with suffix .par in the following order:

```
  curve1
$fwd ... $
$init ... $
  curve2
$fwd ... $
$init ...$
<repeat: title,$fwd...$,$init...$ for other curves>
```

The null string "$fwd $" or "$init $" (without the ") must be used if no parameters are being changed from the preceding curve. Remember, namelist parameters are only needed to override previous values (by default or otherwise set). An example .par file (e.g., test1.par assumed stored in ~/p/fwdinf/test1.par) illustrates several of these points:

```
fwdinf/test1: vary sig(2)
$fwd mm=2,mode=1
sig=.001,.3, h= 100, shift=-1,
x1=.1,nx=5,xm=1000,
x2= 2,x3= 2,x4= 1000,
iplt=1,xt='frequency (hz.)',yt='phase Hz (deg.)'$
```

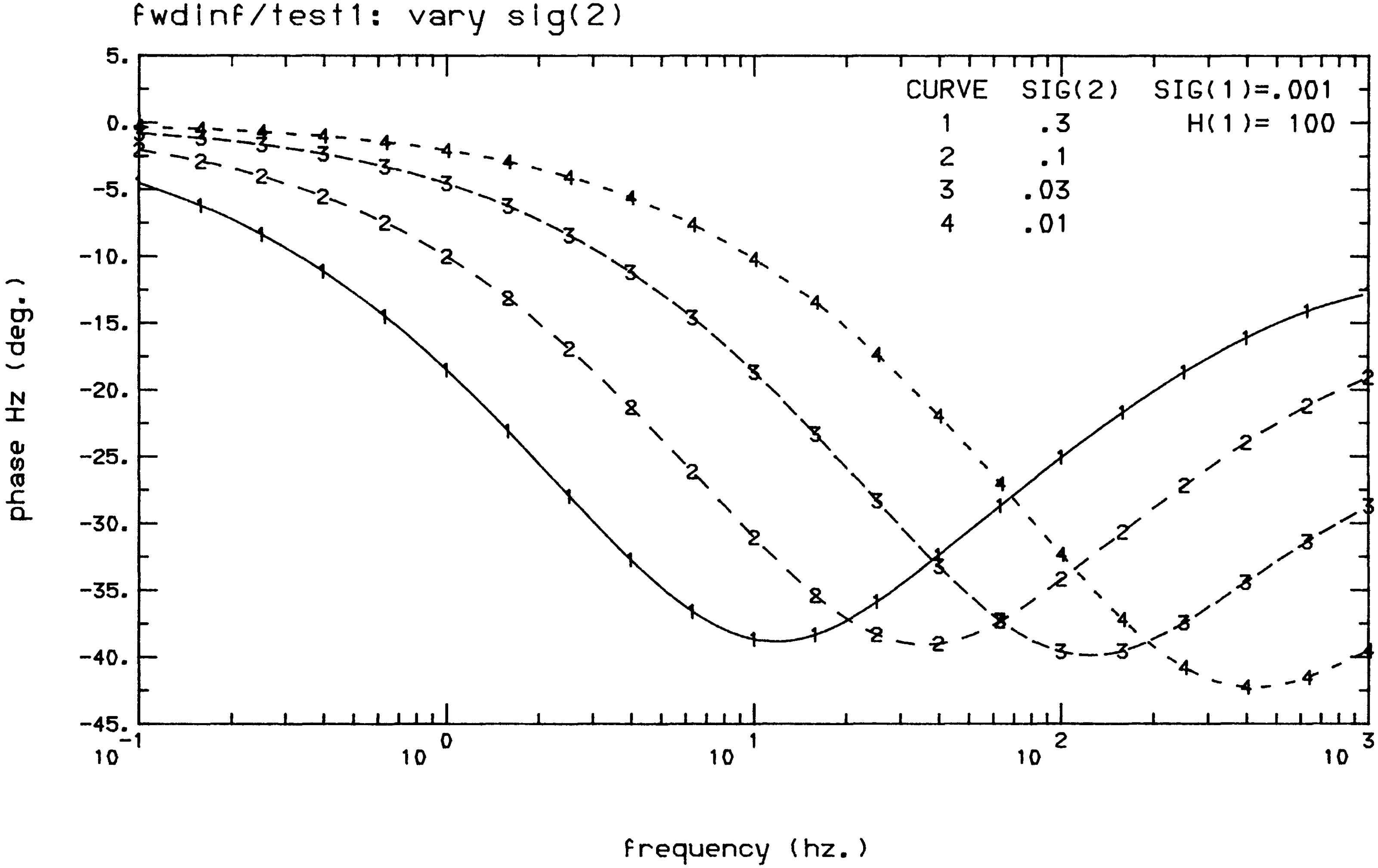
```
$init mm=2,z=-100,iair=0,iall= 1$
-
$fwd sig(2)=.1$
$init $
-
$fwd sig(2)=.03$
$init $
-
$fwd sig(2)=.01$
$init $
```

To execute program FWDINF (assumed executable is ~/lib/fwdinf), the run.com described above for use with NLSINF can be used in the same fashion. Since the Unix directory ~/p/fwdinf contains the parameter file testl.par listed above, then program FWDINF can be executed simply by typing:

```
run fwdinf testl
```

The output files will have suffixes .out, .plt (if $fwd iplt=1), .all (if $init iall=1), and .log; the format of these files are different from the inverse program NLSINF, but more details and examples can be found in Anderson (1984, p.9, and p.13-16). For the above example, $init iall=1 was selected, which creates a special file with suffix .all, and in this case contains: unnormalized amplitude, phase, real-, and imaginary-parts of Hz at each x(i,1) generated from xl to xm (nx>0 points per log decade). As mentioned in the $init definitions, the iall=1 option should never be used with the inversion program (NLSINF).

The plot .plt file (exact format described by Anderson, 1984, p.16-17) will contain 4-curves stacked in the order shown above with parameter sig(2) changed for each curve. The FWDINF program (or any FWD-program) should be used with only a single $fwd or $init parameter varying from curve to curve. Using an external plot routine, the testl example produced the following plot:


fwdinf/test1: vary sig(2)
CURVE SIG(2) SIG(1)=.001
1 .3 H(1)= 100
2 .1
3 .03
4 .01
5.
0.
-5.
-10.
-15.
-20.
-25.
-30.
-35.
-40.
-45.
phase Hz (deg.)
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
frequency (hz.)

Appendix 2.-- Conversion to other systems

Programs NLSINF and FWDINF, and all associated subprograms, were written in "extended" ANSI-standard Fortran-77 for an HP-9000/Unix system. Conversion to other systems without certain standard or non-standard features might necessitate extensive changes to some (or all) of the following items:

(1) Variable names with more than 6-characters.

(2) Character strings delimited by single-quote characters (e.g., 'string'); also, character string concatenation (e.g., 'string1'//'string2'). Some systems might require double-quotes (") for string delimiters.

(3) Non-ANSI "encode" statements of the form: "encode(count,format,string) variables". If used, this statement is to transfer variables to internal string storage via a format specifier having an exact character count. Some f77 systems will allow an internal write statement to strings of the form: "write(string,format) variables", while other systems may not allow either non-ANSI types.

(4) Suppression of arithmetic or exponential underflow messages; also note that a result must be automatically set to 0.0 after any underflow, which is assumed for this program package. If the target system does not set underflows to 0.0, and suppress warning messages, then a suitable conversion procedure must be used for proper operation of this program package.

(5) Non-ANSI "namelist" input and output statements. This could be a major modification or restructure of the entire input scheme. In this case, it would be better to use a "namelist simulator" replacement, if available. Watch out for the preprocessing of namelist input in subprogram PRENAM if making changes.

(6) Non-ANSI "do i=" and "enddo" statements. These can be replaced respectively by "do xx i=" and "xx continue" statements, where xx is any number not already used.

(7) Passed parameter to the "program name (parm)" statement; this statement is used by the run.com line: ~/lib/$1 $2, where the second passed parameter is the file id. If a parameter cannot be passed via the program statement in this way, then some other means must be used in the main programs (NLSINF and FWDINF) to open the input/output files. Note that the present method avoids any prompts

and replies to run NLSINF or FWDINF.

(8) Replacement of machine-dependent constants in subprogram RMDCON (called by program NLSINF) which is currently set for a HP-9000 32-bit machine. See Dennis and others (1979, p.37-38) for a discussion of constants BIG, ETA, and MACHEP; also see comments in the source code for RMDCON and IMDCON.

(9) The "-K" f77 compiler switch was used, which on the HP-9000 will automatically SAVE all local variables in all subprograms. A save statement may need to be inserted in certain subprograms for systems not having a similar f77 compiler option.

(10) Some f77 compilers may flag variables in type or common statements that were declared but not initialized; an f77 switch can usually be set to ignore such warning messages, however, one should not remove these variables in any common blocks if they are used in other subprograms. Some variables were declared for future versions, and the user should not be concerned with "variable not used or initialized" messages.

Appendix 3.-- Inverse program NLSINF i/o example

The following input files (test.par and test.dat) were used to run the inversion program NLSINF on an HP-9000 system. (This data was generated from the associated forward program FWDINF using a known model, and with input as described in Appendix 1; random errors were not added to the generated data, so that the near-exact model would be recovered.)

The corresponding NLSINF output file (test.out) is listed immediately following the input files. File test.out was also used to plot the final observed (obs.) and calculated (cal) curves using an external plotter. The symbol "0" represents the observed y(i) in the plot, and the solid line connects the least-squares calculated (cal) curve for the final layered model solution. In this case, near zero residules (res) were obtained as expected. More realistic "observed data" could be simulated by adding uniform random errors to y(i), etc.

test.par

```
nlsinf/test at 60 hz.
$parms n=42,m=4,sp=3,iprt=-1,
k= 4,ip= 1,ib= 4,
ider=1,iwt=2,
niter= 40,
bl=2*.1e-7,1*5 -1e5,
bh=2*10,1*3000, 1e5,
b=.004,.05,200,-1$
(5g16.8,t1,g16.8)
$init mm= 2,z= -100,iair= 0,iall= 0$
```

test.dat

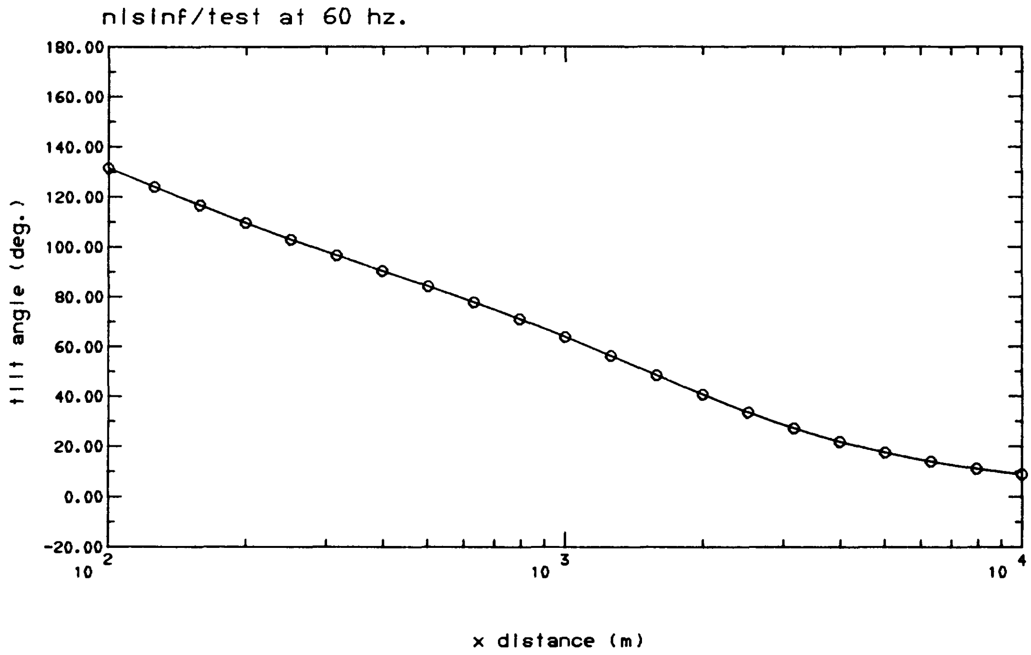
```
  131.46255         100.00000       3.0000000       6.0000000       -60.000000
  124.01151         125.89255       3.0000000       6.0000000       -60.000000
  116.65614         158.48933       3.0000000       6.0000000       -60.000000
  109.59206         199.52626       3.0000000       6.0000000       -60.000000
  102.88794         251.18869       3.0000000       6.0000000       -60.000000
  96.495605         316.22784       3.0000000       6.0000000       -60.000000
  90.284126         398.10727       3.0000000       6.0000000       -60.000000
  84.077484         501.18738       3.0000000       6.0000000       -60.000000
  77.690880         630.95752       3.0000000       6.0000000       -60.000000
  70.965714         794.32849       3.0000000       6.0000000       -60.000000
  63.807411         1000.0004       3.0000000       6.0000000       -60.000000
  56.229080         1258.9259       3.0000000       6.0000000       -60.000000
  48.398746         1584.8938       3.0000000       6.0000000       -60.000000
  40.653133         1995.2631       3.0000000       6.0000000       -60.000000
  33.439922         2511.8875       3.0000000       6.0000000       -60.000000
  27.136520         3162.2791       3.0000000       6.0000000       -60.000000
  21.860031         3981.0735       3.0000000       6.0000000       -60.000000
  17.516603         5011.8745       3.0000000       6.0000000       -60.000000
  13.982637         6309.5762       3.0000000       6.0000000       -60.000000
  11.137686         7943.2861       3.0000000       6.0000000       -60.000000
  8.8612442         10000.005       3.0000000       6.0000000       -60.000000
-.23043539E-01      100.00000       3.0000000       7.0000000       -60.000000
-.28838959E-01      125.89255       3.0000000       7.0000000       -60.000000
-.35995036E-01      158.48933       3.0000000       7.0000000       -60.000000
-.44762131E-01      199.52626       3.0000000       7.0000000       -60.000000
-.55393144E-01      251.18869       3.0000000       7.0000000       -60.000000
-.68107925E-01      316.22784       3.0000000       7.0000000       -60.000000
-.83029591E-01      398.10727       3.0000000       7.0000000       -60.000000
-.10007871          501.18738       3.0000000       7.0000000       -60.000000
-.11880919          630.95752       3.0000000       7.0000000       -60.000000
-.13818903          794.32849       3.0000000       7.0000000       -60.000000
-.15637954          1000.0004       3.0000000       7.0000000       -60.000000
-.17066576          1258.9259       3.0000000       7.0000000       -60.000000
-.17778558          1584.8938       3.0000000       7.0000000       -60.000000
-.17503601          1995.2631       3.0000000       7.0000000       -60.000000
-.16211365          2511.8875       3.0000000       7.0000000       -60.000000
-.14230698          3162.2791       3.0000000       7.0000000       -60.000000
-.12056536          3981.0735       3.0000000       7.0000000       -60.000000
-.99946596E-01      5011.8745       3.0000000       7.0000000       -60.000000
-.81558190E-01      6309.5762       3.0000000       7.0000000       -60.000000
-.65886565E-01      7943.2861       3.0000000       7.0000000       -60.000000
-.52886307E-01      10000.005       3.0000000       7.0000000       -60.000000
```

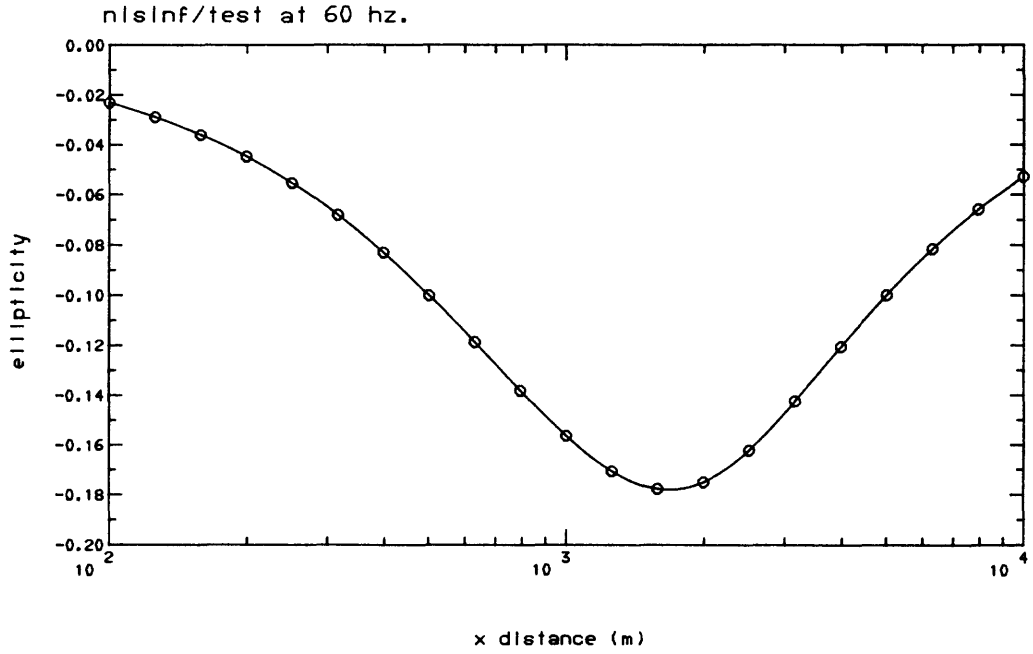
test.out

```
<nlsinf>:     nlsinf/test at 60 hz.

$init
mm =  2,
z =  -100.0,
tol =  1.000000E-09,
iair =  0,
iall =  0
$end


parameter order--

  b( 1) = sigma( 1)
  b( 2) = sigma( 2)
  b( 3) = thick( 1)
  b( 4) = shift parameter in: b( 4)*y(i)


{nlsol}:        nlsinf/test at 60 hz.

n=         42    k=          4    ip=         1    m=          4    ialt=      10
istop=      1    iwt=        2    ider=       1    iprt=      -1    niter=     40
iout=       1    sp=         3

parameters held fixed: ib=  4

fmt=(5g16.8,t1,g16.8)
```

```
parameter lower bounds: bl=

  .99999999E-08   .99999999E-08   .50000000E+01  -.10000000E+06

initial parameters: b=

  .40000002E-02   .50000001E-01   .20000000E+03  -.10000000E+01

parameter higher bounds: bh=

  .10000000E+02   .10000000E+02   .30000000E+04   .10000000E+06

parameter index:  1  2  3  4
reordered as...:  1  2  3

reordered parameters:

  .40000002E-02   .50000001E-01   .20000000E+03

** nlitr (ider=0) or nl2sno (ider=1) called:  1 **


     i     initial x(i)       d(i)

     1      .200013E-01      .618E+02
     2      .707697E-01      .637E+02
     3      .258017E+00      .345E+02

    it     nf       f          df         cosmax       var


     0      1    .341E+02               .985E+00
     1      3    .753E+01    .266E+02   .998E+00   .246E+02
     2      4    .119E-01    .752E+01   .787E+00   .389E+02
     3      5    .133E-02    .106E-01   .919E+00   .375E+02
     4      6    .122E-03    .121E-02   .958E+00   .378E+02
     5      7    .238E-06    .122E-03   .683E+00   .387E+02
     6      8    .105E-07    .227E-06   .616E+00   .356E+02
     7      9    .238E-08    .808E-08   .449E+00   .224E+02
     8     10    .445E-09    .193E-08   .304E+00   .260E+02
     9     11    .445E-09   -.235E-09   .304E+00   .464E+01

***** x-convergence *****


function       .444599D-09  variability    .464326E+01
func. evals       11        grad. evals       9
grad. norm     .154237E-02  cosmax         .304016E+00

     i      final x(i)        d(i)          g(i)

     1     -.999934E-02      .153E+03     -.139E-02
     2      .447349E-01      .910E+02      .599E-03
     3      .418659E+00      .342E+02     -.294E-03


covariance = scale * (j**t * j)**-1

row  1     .7902E-13
row  2    -.1231E-12    .2146E-12
row  3    -.6415E-12    .1056E-11   .5369E-11
```

```
  i   obs.y(i)       cal           res        %res.err       x(i,1)        x(i,2)        x(i,3)        x(i,4)        wt(i)
  1  .131463E+03   .131462E+03   .610E-04   .464278E-04   .100000E+03   .300000E+01   .600000E+01  -.600000E+02   .760673E-02
  2  .124012E+03   .124011E+03   .458E-04   .369130E-04   .125893E+03   .300000E+01   .600000E+01  -.600000E+02   .806377E-02
  3  .116656E+03   .116656E+03   .305E-04   .261603E-04   .158489E+03   .300000E+01   .600000E+01  -.600000E+02   .857220E-02
  4  .109592E+03   .109592E+03   .763E-05   .696163E-05   .199526E+03   .300000E+01   .600000E+01  -.600000E+02   .912475E-02
  5  .102888E+03   .102888E+03   .153E-04   .148305E-04   .251189E+03   .300000E+01   .600000E+01  -.600000E+02   .971931E-02
  6  .964956E+02   .964956E+02   .153E-04   .158129E-04   .316228E+03   .300000E+01   .600000E+01  -.600000E+02   .103632E-01
  7  .902841E+02   .902841E+02   .153E-04   .169009E-04   .398107E+03   .300000E+01   .600000E+01  -.600000E+02   .110761E-01
  8  .840775E+02   .840775E+02   .000E+00   .000000E+00   .501187E+03   .300000E+01   .600000E+01  -.600000E+02   .118938E-01
  9  .776909E+02   .776909E+02  -.229E-04  -.294606E-04   .630958E+03   .300000E+01   .600000E+01  -.600000E+02   .128715E-01
 10  .709657E+02   .709657E+02  -.229E-04  -.322524E-04   .794328E+03   .300000E+01   .600000E+01  -.600000E+02   .140913E-01
 11  .638074E+02   .638074E+02  -.114E-04  -.179354E-04   .100000E+04   .300000E+01   .600000E+01  -.600000E+02   .156722E-01
 12  .562291E+02   .562291E+02  -.381E-05  -.678421E-05   .125893E+04   .300000E+01   .600000E+01  -.600000E+02   .177844E-01
 13  .483987E+02   .483987E+02   .763E-05   .157636E-04   .158489E+04   .300000E+01   .600000E+01  -.600000E+02   .206617E-01
 14  .406531E+02   .406531E+02   .343E-04   .844518E-04   .199526E+04   .300000E+01   .600000E+01  -.600000E+02   .245984E-01
 15  .334399E+02   .334399E+02   .191E-04   .570381E-04   .251189E+04   .300000E+01   .600000E+01  -.600000E+02   .299044E-01
 16  .271365E+02   .271365E+02   .324E-04   .119488E-03   .316228E+04   .300000E+01   .600000E+01  -.600000E+02   .368507E-01
 17  .218600E+02   .218600E+02   .381E-05   .174506E-04   .398107E+04   .300000E+01   .600000E+01  -.600000E+02   .457456E-01
 18  .175166E+02   .175166E+02   .172E-04   .979993E-04   .501187E+04   .300000E+01   .600000E+01  -.600000E+02   .570887E-01
 19  .139826E+02   .139826E+02   .191E-04   .136409E-03   .630958E+04   .300000E+01   .600000E+01  -.600000E+02   .715173E-01
 20  .111377E+02   .111377E+02  -.458E-04  -.411003E-03   .794329E+04   .300000E+01   .600000E+01  -.600000E+02   .897853E-01
 21  .886124E+01   .886118E+01   .620E-04   .699555E-03   .100000E+05   .300000E+01   .600000E+01  -.600000E+02   .112851E+00
 22 -.230435E-01  -.230434E-01  -.142E-06  -.614324E-03   .100000E+03   .300000E+01   .700000E+01  -.600000E+02   .433961E+02
 23 -.288390E-01  -.288386E-01  -.389E-06  -.134990E-02   .125893E+03   .300000E+01   .700000E+01  -.600000E+02   .346753E+02
 24 -.359950E-01  -.359945E-01  -.570E-06  -.158349E-02   .158489E+03   .300000E+01   .700000E+01  -.600000E+02   .277816E+02
 25 -.447621E-01  -.447616E-01  -.566E-06  -.126502E-02   .199526E+03   .300000E+01   .700000E+01  -.600000E+02   .223403E+02
 26 -.553931E-01  -.553926E-01  -.581E-06  -.104914E-02   .251189E+03   .300000E+01   .700000E+01  -.600000E+02   .180528E+02
 27 -.681079E-01  -.681073E-01  -.663E-06  -.973614E-03   .316228E+03   .300000E+01   .700000E+01  -.600000E+02   .146826E+02
 28 -.830296E-01  -.830290E-01  -.596E-06  -.717877E-03   .398107E+03   .300000E+01   .700000E+01  -.600000E+02   .120439E+02
 29 -.100079E+00  -.100078E+00  -.559E-06  -.558357E-03   .501187E+03   .300000E+01   .700000E+01  -.600000E+02   .999214E+01
 30 -.118809E+00  -.118809E+00  -.395E-06  -.332367E-03   .630958E+03   .300000E+01   .700000E+01  -.600000E+02   .841686E+01
 31 -.138189E+00  -.138189E+00   .447E-07   .323495E-04   .794328E+03   .300000E+01   .700000E+01  -.600000E+02   .723646E+01
 32 -.156380E+00  -.156380E+00   .373E-06   .238221E-03   .100000E+04   .300000E+01   .700000E+01  -.600000E+02   .639470E+01
 33 -.170666E+00  -.170666E+00   .685E-06   .401633E-03   .125893E+04   .300000E+01   .700000E+01  -.600000E+02   .585941E+01
 34 -.177786E+00  -.177786E+00   .849E-06   .477745E-03   .158489E+04   .300000E+01   .700000E+01  -.600000E+02   .562475E+01
 35 -.175036E+00  -.175037E+00   .834E-06   .476737E-03   .199526E+04   .300000E+01   .700000E+01  -.600000E+02   .571311E+01
 36 -.162114E+00  -.162114E+00   .641E-06   .395246E-03   .251189E+04   .300000E+01   .700000E+01  -.600000E+02   .616851E+01
 37 -.142307E+00  -.142307E+00   .373E-06   .261778E-03   .316228E+04   .300000E+01   .700000E+01  -.600000E+02   .702706E+01
 38 -.120565E+00  -.120566E+00   .149E-06   .123594E-03   .398107E+04   .300000E+01   .700000E+01  -.600000E+02   .829426E+01
 39 -.999466E-01  -.999466E-01  -.149E-07  -.149091E-04   .501187E+04   .300000E+01   .700000E+01  -.600000E+02   .100053E+02
 40 -.815582E-01  -.815582E-01   .373E-07   .456765E-04   .630958E+04   .300000E+01   .700000E+01  -.600000E+02   .122612E+02
 41 -.658866E-01  -.658859E-01  -.656E-06  -.995131E-03   .794329E+04   .300000E+01   .700000E+01  -.600000E+02   .151776E+02
 42 -.528863E-01  -.528861E-01  -.246E-06  -.464904E-03   .100000E+05   .300000E+01   .700000E+01  -.600000E+02   .189085E+02

** rmserr=   .21465990E-04  ave|%res.err|=   .339562E-03

correlation matrix
 1   .1000E+01
 2  -.9455E+00   .1000E+01
 3  -.9848E+00   .9841E+00   .1000E+01

  **parm_sol.   std_error   rel_error   % error **

 1   .9998E-03   .2811E-06  -.2811E-04  -.2811E-02
 2   .2000E-01   .4632E-06   .1035E-04   .1035E-02
 3   .5000E+03   .2317E-05   .5535E-05   .5535E-03


********  e n d  ********    nlsinf/test at 60 hz.

parameter name      final solution        resistivity   layer depth

   1   sigma( 1) =    .99984417E-03   1   .10001559E+04
   2   sigma( 2) =    .19998765E-01   2   .50003090E+02
   3   thick( 1) =    .49998630E+03                         1    .49998630E+03
   4 * shift     =   -.10000000E+01

     * fixed
```


nlsinf/test at 60 hz.
180.00
160.00
140.00
120.00
100.00
80.00
60.00
40.00
20.00
0.00
-20.00
tilt angle (deg.)
10 2
10 3
10 4
x distance (m)


nlsinf/test at 60 hz.
0.00
-0.02
-0.04
-0.06
-0.08
-0.10
-0.12
-0.14
-0.16
-0.18
-0.20
ellipticity
10 2
10 3
10 4
x distance (m)

Appendix 4.-- Source code availability

The current version of the source code (and test files) may be obtained by writing directly to the author (U.S. Geological Survey, Mail Stop 964, Box 25046, Denver Federal Center, Denver, CO., 80225), and enclosing a magnetic tape to be copied and returned. This method of releasing the source code was selected in order to satisfy requests for the latest (possibly updated) version. The attached list gives the names of the programs and associated subprograms included in each file on the distributed tape. New versions may contain replacement or additional subprograms as required, and will be noted in a distribution note with the tape copy.

Unless otherwise requested, the magnetic tape will be recorded in the following mode:

> Industry compatible: 9-track, standard ANSI-labeled, ASCII-mode, odd-parity, 1600-bpi density, 80-character records (blocked 50-card images, or 4000-characters, per physical block), and contained on 6 files named: nlsinf.hp, fwdinf.hp, run.com, testl.par, test.par, and test.dat. These files have the following contents:

file 1: nlsinf.hp (left-to-right order)

```
nlsinf (main)  fcode         subz          rtefun
dumypcode      sigsubend     errmsg        errmsgi
minmax         nlsol2        nonblank      polar2
prenam         tilt10        zflags        zfour0
zfour1         zlagf0        zlagf1        asinh
calcr          dfault        erf           erfinv
imdcon         intran        loc           nl2sno
nlitr          rmdcon        splinl        spoint
tcheb          vscopy        warn          itsmry
nl2itr         parchk        qapply        rptmul
slupdt         slvmul        stopx         v2norm
vaxpy          vcopy         assess        covclc
dotprd         dupdat        gqtstp        linvrt
livmul         lmstep        lsqrt         lsvmin
ltsqar         qrfact        setio         clober
```

file 2: fwdinf.hp (left-to-right order)

```
fwdinf (main)  fcode         subz          rtefun
errmsg         errmsgi       fwdsol        minmax
nonblank       polar2        prenam        tilt10
zflags         zfour0        zfour1        zlagf0
zlagf1         splinl        spoint        vecprt
setio          clober
```

file 3: run.com (as listed in this report)
----

file 4: test1.par (for fwdinf in Appendix 1)
----

file 5: test.par (for nlsinf in Appendix 3)
----

file 6: test.dat (for nlsinf in Appendix 3)
