296
5/14
LA-3830

MASTER

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

Predictable Unfolding
in the Time Domain

UNITED STATES
ATOMIC ENERGY COMMISSION
CONTRACT W-7405-ENG. 36

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency Thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or

B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

This report expresses the opinions of the author or authors and does not necessarily reflect the opinions or views of the Los Alamos Scientific Laboratory.

Printed in the United States of America. Available from
Clearinghouse for Federal Scientific and Technical Information
National Bureau of Standards, U. S. Department of Commerce
Springfield, Virginia 22151

Price: Printed Copy \$3.00; Microfiche \$0.65

LOS ALAMOS SCIENTIFIC LABORATORY
of the
University of California
LOS ALAMOS • NEW MEXICO

Report written: September 1967

Report distributed: May 29, 1968

Predictable Unfolding
in the Time Domain

by

E. K. Hodson

LEGAL NOTICE

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:
A. Makes any warranty or representation, expressed or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.
As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, in the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

**THIS PAGE
WAS INTENTIONALLY
LEFT BLANK**

PREDICTABLE UNFOLDING IN THE TIME DOMAIN

By

E. K. Hodson

ABSTRACT

The convolution integral generally lends itself readily to numerical solution. The inverse numerical process, unfolding, frequently produces divergent solutions, and nearly always seriously increases the noise present in the resulting tabular function. Since noise is present in all experimentally derived tables (even sampling of virtually noise-free data may produce noise), we consider this to be a major cause of divergent or oscillatory solutions. The influence of noise on the unfolding process is controlled predictably by the method described here, in which a modified (constrained) inverse of the system response function is folded with the data. To generate the constrained inverse we unfold a properly chosen filter with the system response function, $g(t)$, modifying the latter by truncation of its early portion. When this modified inverse is convolved with the data to be unfolded, noise enhancement is minimal. CONFOLD is a computer program which determines the minimum required truncation of $g(t)$ by iteration, and unfolds a Gaussian filter, producing a constrained inverse which is then folded with the data. The unfolded frequency response is computed by use of the Fourier integral. CONFOLD will also evaluate the frequency response of $g(t)$, making it possible to predict filter characteristics giving the best compromise between bandwidth and noise suppression. The program may be used either for folding or unfolding. Its normal outputs are Stromberg 4020 paper listing and punched cards.

I. METHODS, PROBLEMS, AND REMEDIES

We shall consider some of the problems involved in solving the convolution integral

$$Y(t) = \int_0^t y(\lambda) g(t - \lambda) d\lambda \quad (1)$$

for $y(t)$ in a computer. A method, CONFLD, used in LASL Group J-14, which minimizes these problems will be described.

The functions Y , y , and g appear in the com-

puter as numerical tables and are normally not available in analytic form. Therefore, the following approximation of Eq. 1 is used.

$$N = (t - t_0)/\Delta\lambda,$$

$$Y(t) \approx \sum_{K=1}^N y(K\Delta\lambda) g(t - K\Delta\lambda) \Delta\lambda. \quad (2)$$

We shall consider Y , y , and g as functions of time: $g(t)$ is the response of a real physical system to a δ function, $y(t)$ is an input signal to

the system, and $Y(t)$ is the signal as it appears at the output of the system. If $y(t)$ and $g(t)$ are the input functions and Eq. 2 is solved for $Y(t)$, the process is called a "fold" and is quite stable if $g(t)$ is the impulse response of a physical system.

In most applications, however, $y(t)$ is the unknown quantity and $Y(t)$ and $g(t)$ are the observed functions. Therefore it becomes necessary to solve Eq. 2 for $y(t)$. An approximate solution for $y(t)$ can be written as follows,

$$y(\Delta\lambda) = Y(\Delta\lambda)/[g(\Delta\lambda) \Delta\lambda],$$

$$y(t) \approx \left\{ y(t) - \sum_{K=1}^{N-1} y(K\Delta\lambda) g[t - (K-1)\Delta\lambda] \Delta\lambda \right\} / [g(\Delta\lambda) \Delta\lambda], \quad t > \Delta\lambda, \quad (3)$$

or

$$y(t) \approx \sum_{K=1}^N Y(K\Delta\lambda) \cdot a(t - K\Delta\lambda) \Delta\lambda, \quad (4)$$

where $a(t)$ is the "inverse" of $g(t)$, i.e., a function which would yield a δ function if folded with $g(t)$. This is called "unfolding."

The latter method, Eq. 4, has been considered by Longley.¹ He has outlined some of the problems involved in attempting to invert $g(t)$ if $g(t)$ is a tabular function. The resulting $a(t)$ commonly oscillates divergently above and below zero, frequently reaching values that exceed the limits of a computer. This almost invariably happens if the ratio $g_{\max}/g(\Delta\lambda)$ is sufficiently large. Eq. 3 frequently behaves in the same manner, producing an oscillatory, divergent $y(t)$. It is clear that although we could recover the original $Y(t)$ for a time by folding the divergent $y(t)$ with $g(t)$, the divergent $y(t)$ is not the function that was the original input to the system.

A study of this problem was proposed by Jack Shearer, of Lawrence Radiation Laboratory, at an informal discussion in June 1965. He suggested that noise in the tabular functions Y and g , as well as sampling frequencies, $1/\Delta\lambda$, could contribute substantially to the instabilities inherent in unfolding.

The problem is perhaps more easily visualized in the frequency domain than in the time domain. We can evaluate the following form of the Fourier integral trapezoidally in the computer:

$$G(F) = \int_0^t g(t) \cos(2\pi Ft) dt - i \int_0^t g(t) \sin(2\pi Ft) dt. \quad (5)$$

This gives a $G(F)$ that approximates the frequency response of the system that we have described in time domain by $g(t)$. The ratios of the real and imaginary parts of Eq. 5 are the tangents of the phase angles for each F , so phase can be computed in the same process. Figures 1, 2, and 3 illustrate the results of such an exercise. Figure 1 is a plot of a typical $g(t)$; Fig. 2 is the frequency response, amplitude vs frequency, in MHz; and Fig. 3 is phase in degrees vs frequency. In Fig. 2 we see that the function $G(F)$ is quite smooth and continuous below about 600 MHz. As frequency increases, $G(F)$ becomes erratic and finally breaks up above 4 GHz. A similar effect is observed in the plot of phase vs frequency, Fig. 3; the function displays a number of discontinuities above approximately 700 MHz.

If we assume that a theoretical "unfold" using this typical $g(t)$ may bring all portions of the amplitude transform of the measured function back to unity and return all portions of the phase transform to zero, we may expect trouble in regions of the spectrum where amplitude and phase are not well defined. The exception, of course, is the trivial case where any function unfolded with itself always yields $\delta(t)$.

We may now predict the appearance of oscillatory behavior at some frequency above 700 MHz if this $g(t)$ is inverted or data are unfolded with it. An inverse of an approximation of $g(t)$ appears in Fig. 4. It is, indeed, oscillating with a predominant frequency component of about 2 GHz. It is converging only because we eliminated the early, near zero portion, or "toe" of $g(t)$, before attempting the inversion. This is a modification of the technique proposed by Longley and will be discussed in greater detail later (see Fig. 5).

For this oscillatory inverse to be truly useful in an unfolding process, it should result in something approximating a δ function if it is folded with the measured $g(t)$. This was attempted, and the result is shown in Fig. 6. It is clear that the result

does not resemble a δ function.

The Fourier integrals, Eq. 5, were solved for the function of Fig. 6, and the resulting frequency response and phase shifts are shown in Figs. 7 and 8, respectively. Note that the amplitude and phase curves are both erratic above about 800 MHz.

We can now predict that folding our "inverse" with data that contain even a small amount of noise will produce oscillation at some frequency above 800 MHz. The function of Fig. 9 was folded with this "inverse," and the result is the severely oscillating function of Fig. 10.

Let us now compare the frequency response of $g(t)$, Fig. 2, with that of the "unfolded" $g(t)$, Fig. 7. We see that for most of the region above 700 MHz we have enhanced frequency components by a factor of 100 or more. If we are to have any confidence in the results an exceedingly low noise level both in $g(t)$ and in the data to be unfolded is necessary.

The amplitude and phase curves of Figs. 7 and 8 imply that we have done a passable job of unfolding in the region below about 500 MHz, but the region above 500 MHz leaves much to be desired.

It should be noted that many computer runs involving Eq. 3 show the same oscillatory behavior as is produced by folding the "inverse" of $g(t)$ with the data as in Eq. 4. This implies that the oscillatory behavior is not uniquely associated with the process of calculating an inverse.

For an unfold to be satisfactory the unfolded system frequency response should be predictable.

That is, if we cannot unfold so that the frequency response is unity from $F = 0$ to $F = \infty$, the response should depart from unity in a reasonably smooth and predictable manner.

Let us now define a satisfactory response characteristic. The unfolded system frequency response should be constant in amplitude, i.e., $G_{UF}(F) = 1$, and the phase should equal zero or be linear over a region called the passband. Amplitude should then fall rapidly to the $G(F)$ computed from the measured $g(t)$ for the entire region outside the passband, the stop band. The passband should be wide enough to pass the applied signal with minimum and acceptable distortion. That is, the frequency response of the "inverse," $G_{INV}(F)$, should equal $1/G(F)$ for all values of F in the pass band and

should equal unity in the stop band.

The above conditions are not attainable; however, we can do well with an approximation. Fig. 11 shows $G_{INV}(F)$ or $1/G(F)$ for the measured $g(t)$. If we multiply $G_{INV}(F)$ by the transform of an appropriately chosen filter we obtain a modified or constrained version of $G_{INV}(F)$. This, which we shall call $G_c(F)$, is illustrated in Fig. 12. The filter used in this case was a Gaussian filter of the form:

$$U(t) = \left[1/\sqrt{2\pi} \right] e^{-0.5[(t/\tau) - (1/D\tau)]^2}, \quad (6)$$

where $1/D$ is a delay term: $\tau = 132/F_{3db}$. F_{3db} is the frequency in MHz whose value is -3db in the frequency domain transform of Eq. 6. The frequency domain characteristics of this filter are that phase is linear (looks like a delay term) and the frequency response in db is proportional to the frequency squared.

From Fig. 12 we can see that portions of the spectrum from about 700 MHz to 1.2 GHz are still enhanced by a factor of 50 to 90. If much noise energy, from $g(t)$ or $Y(t)$, exists in this portion of the spectrum, our inverse or unfold may still be oscillatory. This implies that the filter needs to be more effective at these frequencies. A different filter equation might accomplish this; the removal of a portion of the toe of $g(t)$ will also depress these frequency components.

If we compute $G_{INV}(F)$ for the modified $g(t)$ of Fig. 5 and multiply by the transform of our Gaussian filter, we have the transform of Fig. 13 which duplicates the original inverse transform reasonably well to slightly above 400 MHz. Note that no frequency components are enhanced by much more than a factor of 10. If the noise component in this portion of the spectrum is a few percent or less, we should not be troubled by oscillations in the answer, $y(t)$, assuming we have not excessively distorted phase in the passband by removing a portion of the toe of $g(t)$. It is true that we have depressed frequency components over much of the stop band. It is not obvious that real damage to $y(t)$ will result if the passband is sufficiently wide. This was stated as one of the initial conditions for the filter frequency response.

Let us now consider how we may compute a modified inverse of the measured $g(t)$ in the time

domain whose frequency response looks like the modified inverse transform of Fig. 13. Let $g'(t)$ be the function $g(t)$ with the toe removed and $U(t)$ be our Gaussian filter of Eq. 6. Using Eq. 3 we can solve for a "constrained" inverse, $a_c(t)$, as follows:

$$a_c(t) \approx \left\{ \begin{array}{l} U(t) - \sum_{K=1}^{N-1} a_c(K\Delta\lambda) g'[t-(K-1)\Delta\lambda]\Delta\lambda \\ /g'(\Delta\lambda)\Delta\lambda, \quad t > \Delta\lambda. \end{array} \right\} \quad (7)$$

The frequency response of $a_c(t)$ appears in Fig. 14. As we might expect, it closely resembles that of Fig. 13.

The inverse of $g'(t)$ computed as suggested by Longley² can also be folded with $U(t)$ to yield $a_c(t)$. If we now fold $a_c(t)$ with $Y(t)$, Eq. 4, we have $y_c(t)$. This is equivalent to folding $Y(t)$ with the inverse of $g'(t)$ and then folding the result with $U(t)$.

There are several advantages in computing $a_c(t)$ as in Eq. 7. There are fewer computations in the computer, and the relative values of the sums of the products are less dissimilar; therefore, the accuracy may be somewhat improved. Another advantage is that $a_c(t)$ can be tested for convergence more easily than the oscillating inverse of $g'(t)$, shown in Fig. 4.

Let us see what happens when we fold $a_c(t)$ with the measured $g(t)$ of Fig. 1. The result is the unfolded impulse response of Fig. 15. The function of Fig. 15 is still not a δ function, but it is a reasonable approximation of the Gaussian filter, $U(t)$. If we fold $a_c(t)$ with the noisy data of Fig. 9, we get the reasonably smooth overplotted function of Fig. 10, a considerable improvement over the function resulting from the fold with the inverse of $g'(t)$.

Figure 16(a) is a composite which illustrates first a fold of a function with the measured $g(t)$, then an unfold using Eq. 4 and the constrained inverse, $a_c(t)$, which demonstrates a near recovery of the original function. Function C is the result of the fold, B is the result of the unfold, and A is the input function. Note that B is delayed in time with respect to A due largely to our having removed the

toe of $g(t)$. In Fig. 16(b) we see B time-shifted to overlie A, to facilitate comparison.

In all unfolding operations where $Y(t)$ does not originate at zero, a convergent inverse is desirable. Any such function should be "front-fitted" long enough for the inverse to decay to values such that little or no integral remains. Otherwise an erroneous level shift may be introduced into the unfolded answer. This is strictly true in a computer program that does not require the table $a_c(t)$ to be the same length as $Y(t)$. In any event, if $a_c(t)$ is continually increasing in magnitude, "front-fitting" errors are propagated into "current" time.

If $Y(t)$ originates and ends in a time so short that $a_c(t)$ does not grow large enough to interfere with computer accuracy, a convergent inverse is not required.

II. CONFLD, THE APPLIED TECHNIQUE

We will now discuss how we treat the problem in the CONFLD computer program.

Most of the tables of $g(t)$ used for unfolding result from field measurements. The errors in these measurements are not yet completely understood. We therefore assume the data to be good for a dynamic range of 50. This implies that the first 2% of the integral of $g(t)$ is outside that range. Acting upon this assumption, we eliminate all points in the toe of $g(t)$ whose integrals do not accumulate to equal 2% of the total integral of $g(t)$. We then normalize the remaining $g(t)$ so that the remaining integral equals 1.0. Following an equal partitioning of $g'(t)$ by linear interpolation we have our first table, which we will apply to Eq. 7 in an attempt to unfold $U(t)$.

At this point we must consider the parameters for our filter function, Eq. 6. The parameter, F3db, is an input frequency in MHz, which determines the frequency at which the frequency response of $U(t)$ is equal to -3db. The frequency normally chosen for this input is down to about -20 db on the frequency response curve computed from the measured $g(t)$. t is common to both $U(t)$ and $g'(t)$, and Δt is chosen such that there will be at least 15 samples per cycle at F3db MHz.

We now attempt an unfold of $U(t)$ with $g'(t)$. As t progresses toward t_{max} , we test the resulting

values of $a_c(t)$ to determine if convergence to a satisfactorily small value at $t_{\max}$ is implied. If the convergence test is satisfied, we continue the computation to $t_{\max}$. The resulting $a_c(t)$ is then stored for later evaluation and use.

If at any time during the preceding computation the convergence test is not satisfied, the computation is terminated and a new $g'(t)$ is computed in the following manner. From the original computation of $g'(t)$ we evaluate $\Delta g = (g_{\max} - g'_0)/20$. A new g'_0 is computed by adding Δg to g'_0 , and all $g'(t)$'s in the toe of the function which are less than this value are eliminated. A t is computed for this value by linear interpolation. After equally partitioning the remainder of the table by linear interpolation, we again attempt an unfold of $U(t)$ observing the above convergence test on $a_c(t)$.

If $a_c(t)$ is not convergent, we again increase the value of g'_0 by Δg and proceed as before. This process of eliminating portions of the toe of $g'(t)$ is graphically illustrated in Fig. 17. A maximum of 20 iterations would of course eliminate all points prior to $g'_{\max}$. Early convergence usually occurs and is much to be desired. This iterative process with its accompanying tests is entirely automatic, and the final ratio, $g'_{\max}/g'_0$, is printed out.

The resulting convergent $a_c(t)$ is now folded with an equally spaced version of the measured $g(t)$. The resulting function, the unfolded impulse response, is then evaluated for frequency response and phase distortion. The frequency response should approximate that of $U(t)$ in the passband. Phase distortion should be minimal at frequencies below F3db. Figure 18 shows comparison of the frequency response curves of $U(t)$ and the unfolded impulse response of the test case. Figure 19 is a plot of the phase characteristics.

Thus far we have assumed $g(t)$ to be the time derivative of a step response resulting from the influence of a system upon a perfect unit step. Unfortunately perfect step generators do not exist. Let us call the time derivative of the function created by a real physical step generator $P(t)$. Since $P(t)$ results from a convolution of $P(t)$ with $\delta(t)$, $P(t)$ appears as an error in the measured $g(t)$. Let $S(t)$ be the impulse response we would have measured

if a real δ function were available with which to pulse the system. The $g(t)$ we actually measured can now be described as follows.

$$g(t) = \int_0^t S(\lambda) P(t-\lambda) d\lambda$$

It follows that the $a_c(t)$ we have considered thus far really consists of a convolution of the inverses of $S(t)$ and $P(t)$ with the filter.

Since any function folded with its theoretical inverse yields a δ function we can remove most of the effect of the step generator from $a_c(t)$ by a convolution of $a_c(t)$ with $P(t)$. This option is provided in CONFLD for those cases where $P(t)$ does not satisfactorily approximate $\delta(t)$. $a_c(t)$ is now ready to be folded with the data. As previously stated a fold or unfold has no real meaning until the convolution has progressed into the data for a long enough time for $g(t)$ or $a_c(t)$ to decay to values near zero, unless the function equals zero for values prior to t_0 . If it is necessary to retrieve answers which exist earlier in time than this, front-fitting is necessary. The type of front-fitting equation will depend upon the nature of the data being unfolded. A linear function is included in CONFLD. This should be replaced by another form of equation when necessary.

A complete sample listing is provided in the appendix.

SUMMARY

There are disadvantages as well as advantages associated with this technique. However, first consider some of the assets. The program is automatic; i.e. it needs no operator control to find the greatest acceptable ratio of $g'_{\max}/g'_0$. It presents an evaluation of the unfolded system in frequency domain for the operators' examination. A number of different experiments can be unfolded to the same band width for valid comparisons. Data noise "blow-up" can be controlled by a proper choice of bandwidth. Phase distortion in the passband is minimal.

Among the disadvantages, we find the following. The unfolded system band width is usually somewhat narrower than that predicted by the Gaussian filter. This is caused by our removing a portion of the toe

of $g(t)$ which also causes the small residual phase distortion in the passband. Systems with excessive phase distortion do not unfold so well as might be desired. Highpass or bandpass systems are special cases which do not unfold with this method so well as do lowpass systems.

The author has, nearing completion, a technique which thus far promises to overcome most of the above objections.

ACKNOWLEDGMENTS

I acknowledge valuable suggestions from R. Humphrey and H. Kruse, during the development of this program. Several of the output-input sub-

routines are modified versions of those originated by Humphrey. Also, frequent discussions with L. Zongker have aided materially in development of the method used in computing the Fourier integral. The efforts of D. Thayer, D. Westervelt, and R. Partridge in the editing of this paper are acknowledged with thanks.

REFERENCES

1. H. J. Longley, "Numerical Solutions and Applications of the Fold Integral," Los Alamos Scientific Laboratory report LA-2729, July 1962, pp. 24-28.
2. Ibid., p. 25.

Fig. 1 System impulse response

Fig. 2 Frequency response of measured $g(t)$

Fig. 3 Phase characteristics of measured $g(t)$

Fig. 4 Inverse of $g(t)$, constrained with truncation of toe

Fig. 5 Measured $g(t)$ with toe removed

Fig. 6 Fold of inverse shown in Fig. 4 with measured $g(t)$

Fig. 7 Frequency response of function shown in Fig. 6

Fig. 8 Nonlinear component of phase shift of function shown in Fig. 6

Fig. 9 Function with superimposed noise

Fig. 10 Unfolds of a noisy table with and without filter constraint

Fig. 11 Frequency response of a theoretical inverse of $g(t)$

Fig. 12 Frequency response of theoretical inverse of $g(t)$
multiplied by frequency response of Gaussian filter

Fig. 13 Frequency response of inverse of $g'(t)$
multiplied by frequency response of Gaussian filter

Fig. 14 Frequency response of the constrained inverse of $g(t)$

Fig. 15 Unfolded impulse response, a fold of $A_c(t)$
with the measured impulse response

Fig. 16(a) Function (C) is the result of a fold of (A) with $g(t)$. (B) results from a fold of (C) with $a_c(t)$

Fig. 16(b) The functions of Fig. 16(a) with (B)
time shifted to overlie (A)

Fig. 17 Truncating the toe of $g(t)$ in increments of Δg

Fig. 18 Frequency response of (A) unfolded impulse response, and (B) Gaussian filter

Fig. 19 Phase characteristics of unfolded impulse response

APPENDIX: CONFLD OPERATIONAL PROCEDURE, IBM 7094

The CONFLD program and data deck setups are shown in Fig. A1. Any number of data sets and problems may be stacked sequentially, limited only by the computer running time requested on the job card.

The first card of the data deck is the title of the data set being processed and is read under format 12A6. Characters appearing in columns 1 through 6 will appear as identification on all output graphs and in the ID field of all punched-card output.

The second card of the data deck is read under format 4E10.0 and contains the following information:

a. Δt , called DELTA, the equal partition increment in nanoseconds, is entered in the first field. This input may be left blank if the fourth entry on this card, F3db, is nonzero.

b. GTMAX, entered in the second field, imposes an upper limit on t for the impulse response, $g(t)$. If no number is entered in this location the program uses the entire table or tables of $g(t)$ as read in.

c. GTEST is entered in the third field. This is an optional override of a portion of the program. It imposes an arbitrary upper limit on the ratio $\epsilon_{\max}/g(\Delta\lambda)$ used in the first approximation of $g(t)$; i.e. $\epsilon_{\max}/g'(\Delta\lambda) \leq \text{GTEST}$. This input is normally left blank.

d. F3db is entered in the fourth field. This number is in MHz and is obtained from the frequency transform of the table or tables of $g(t)$. This number is usually that frequency, F , at which $G(F) \approx -20\text{db}$. If F3db is left blank and IFLD = 0, the program attempts a normal matrix inversion of $g(t)$.

The third card of the data deck is read under format 8I5 and contains the following information:

a. NGT, entered in the first field, is the number of tables, $g(t)$, to be read in with this data set. Any number of tables may be read in and successively convolved such that the final $g(t)$ is the impulse response of a system containing those subsystems from which the individual tables, $g(t)$, originated. If NGT is left blank, the program expects a single table, $g(t)$.

b. IFGF, entered in the second field, is an integer, 1 through 4, which gives the program format

instructions for the data cards, $g(t)$, to be read in.

If IFGF = 1 or 3, the program expects a table of t 's, one blank card, and a table of g 's followed by a blank card with number 1 meaning format 6E12 or 6F12 and number 3 meaning format 7E10 or 7F10.

If IFGF = 2 or 4, the program expects a table of g 's followed by a blank card with 2 meaning format 6E12 and 4 meaning format 7E10 or 7F10. Obviously, if no t 's are read in, it is necessary that $g(t)$ be a function of an equal partition of t and that Δt be entered appropriately in the second card of the data deck.

c. IPP, entered in the third field, is an integer, 1 through 4, which gives the program format instruction identical to item b except that the table $g(t)$ represents a measurement of the input step or pulse used in recording the $g(t)$'s of item b. If IPP = 0 or blank, the program does not expect this table and assumes that a unit step or $\delta(t)$ was the input.

d. INTGF is entered in the fourth field. If INTGF ≥ 1 , the program numerically differentiates $g(t)$. This is required if the read-in response function is $h(t)$, a step response. If INTGF < 1 or blank, the program expects $g(t)$ to be an impulse response and does not differentiate the response function.

e. NAVE is entered in the fifth field. If NAVE ≥ 1 the program smooths $g(t)$ by a three-point unequally weighted average. This exercise is sometimes desirable if $g(t)$ is exceptionally noisy.

f. IPUN entered in the sixth field, is an integer, 1 through 4, which gives the program instructions for punching output on cards.

If IPUN = 1 and IFLD ≥ 1 , the punched output is the convolution of all tables of $g(t)$ read in this data set.

If IPUN = 1 and IFLD = 0, the punched output is the inverse, $a(t)$, of the input table or tables of $g(t)$.

If IPUN = 4 and IYP ≥ 1 , the punched output is the unfolded or folded output, of the FIWF subroutine.

The format of the punched cards is 6E12.5, a table of X followed by a blank card and a table of Y followed by a blank card.

If IPUN = 0 or blank, no output is punched on cards.

g. IYP entered in the seventh field, is an integer. This input tells the main program if data follows to be processed in the FIWF subroutine.

If IYP ≥ 1 , the program expects data as input to the FIWF subroutine.

If IYP = 0 or blank, the main program completes all exercises called for to the point of this call and then goes to the next problem.

h. IFLD entered in the eighth field, is an integer.

If IFLD ≥ 1 , the main program computes $s(t) = g(t)$, and when $s(t)$ is convolved with data, a fold rather than an unfold results.

If IFLD = 0, the main program computes $s(t)$ as the inverse of $g(t)$. The resulting convolution with data is an unfold.

If IFLD ≥ 1 , and both F3db and IYP = 0 the program will compute the Fourier integral of $g(t)$, present graphs of both amplitude and phase and then go to the next problem.

Following the third card of the data deck, the $g(t)$ data are entered.

Following the last blank card of the $g(t)$ data, there are two options described below.

a. If IYP = 0 or blank, the end of job card or the first card of the next problem is entered.

b. If IYP ≥ 1 , subroutine FIWF input follows.

The first card of the data set for this subroutine is read under format 3E10.0. XMIN is entered in the first field, XMAX in the second field, and YZERO in the third field. YZERO is a front-fitting constant.

The second card of the data set for this subroutine is read under format 3I5.

IDF is entered in the first field. This is a number defining the format of the data to be read into this subroutine, and it has the same range and definition as the earlier input IFGF.

IFFT, a front-fitting control, is entered in the second field. If IFFT $\neq 0$, the subroutine linearly front fits the data from $Y(XMIN)$ to YZERO.

M is entered in the third field. If M = 0, the subroutine computes an equal partition of the data by linear interpolation.

If M $\neq 0$, the subroutine computes the equal partition by logarithmic interpolation.

Following the second card of the data set for

this subroutine the tabular data, X and Y(X), are entered. The subroutine will read single or composite sets of data. The upper limit on the number of sets for any composite is 151 per composite or the dimension statement which is protected internally as follows. If, upon completion of reading the Jth set of the composite, more than 3199 values of X and Y(X) have been read in, the subroutine does not store the J+1 set in memory, prints an overload comment, and continues with the problem. The end of any set of data, be it single or composite, is identified by the insertion of an extra blank card.

Following the extra blank card of this data set, the end-of-job card or the first card of the next problem is entered.

As stated in the tabular data stacking instructions for this subroutine, composite sets of information may be read in. It is, of course, required that X or Y(X) be common for all sets of the composite; i.e., on the same X scale. The subroutine will sort Y with respect to X after completion of the read exercise, so that X is monotonically increasing.

Fig. A1 Binary deck configuration

A Fortran IV listing of Confld with its associated subroutines follows. Subroutines RDDT, EQSPC and PUNE 12 are adaptations from programs written by R. W. Humphrey. Subroutine SORT was written by W. D. Gould from a flow diagram by D. L. Shell as published in the Communications of the ACM, July 1959.

Immediately following the Fortran listing is shown the computer output for a typical problem.

04/28/00

CONF - EFN SOURCE STATEMENT - IFN(S) -

COMMON TEM(3000),TEM2(3000),G(3000),G2(1000),GF(1000),HEAD(12)		
DIMENSION TEM3(1000),TEM4(1000)		
C		CONF0110
C		CONF0130
1 FORMAT(12A6)		CONF0140
2 FORMAT(4E10.0)		CONF0150
3 FORMAT(8I5)		CONF0160
4 FORMAT(1H1)		CONF0170
C		CONF0180
11 FORMAT(1H1,12A6)		CONF0190
12 FORMAT(1H ,3X6HDELTA=1PE12.4,2X6HGTMAX=E12.4,		CONF0200
12X6HGTTEST=E12.4,2X5HF308=E12.4,/1H)		CONF0210
13 FORMAT(1H ,3X4HNGT=15,2X6HFM1 G=15,2X14HFMT INP PULSE=15,		CONF0220
12X6HINTGF=15,2X5HNAVE=15,2X5HIPUN=15,2X3HYP=15,2X4HIFLD=15,/1H)		CONF0230
14 FORMAT(1H1,10X27HINTERNALLY EVALUATED GTEST=1PE12.4,/1H)		CONF0240
15 FORMAT(1H ,10X17HRECOMPUTED DELTA=1PE12.4,/1H)		CONF0250
16 FORMAT(1H ,10X21HWHAT FORMAT DO I READ,/1H)		CONF0260
17 FORMAT(1H ,10X15HCOMPUTED DELTA=1PE12.4,/1H)		CONF0270
18 FORMAT(1H1,3X37HREAD IN TAB TOO LARGE, TRUNC TAB USED)		
C		CONF0280
21 FORMAT(6E12.5)		CONF0290
22 FORMAT(7E10.5)		CONF0300
C		CONF0310
29 FORMAT(14X,1A6,5X13HINPUT G TABLE)		
30 FORMAT(45X29HFREQ RESP OF IMPULSE RESPONSE)		
31 FORMAT(1H ,10X1HX,8X7HINVERSE,/1H)		
32 FORMAT(1H ,3X1P2E12.4)		CONF0350
33 FORMAT(14X,1A6,5X14HINPUT G TABLE I5)		CONF0360
34 FORMAT(14X,1A6,5X22HFOLD OF INPUT G TABLES)		CONF0370
35 FORMAT(14X,1A6,5X6HGTEST=1PE12.4,/1H)		CONF0380
36 FORMAT(1H ,13X10HADJUSTED G)		CONF0390
38 FORMAT(14X,1A6,5X19HCONSTRAINED INVERSE)		CONF0400
42 FORMAT(14X,1A6,5X11HINPUT PULSE)		CONF0440
48 FORMAT(90X,1A6)		CONF0500
51 FORMAT(14X,1A6,5X25HUNFOLDED IMPULSE RESPONSE)		CONF0530
53 FORMAT(5X,2HCB)		CONF0550
54 FORMAT(14X,1A6,50X10HFREQ IN MC)		CONF0560
55 FORMAT(45X,30HFREQ RESP OF UNFOLDED IMP RESP)		CONF0570
56 FORMAT(4X,3HDEG)		CONF0580
57 FORMAT(43X,32HNON LIN COMPONENT OF PHASE SHIFT)		CONF0590
58 FORMAT(40X,11HPHASE SHIFT)		
C		CONF0600
100 READ (10,1)HEAD		CONF0610 1
READ (10,2)DELTA, GTMAX, GTEST, F308		CONF0620 3
READ (10,3)NGT,IFGF,IPP, INTGF,NAVE,IPUN,IYP,IFLD		CONF0630
C		CONF0640
C		CONF0650 4
WRITE (12,4)		CONF0660 13
WRITE (9,11)HEAD		CONF0670 14
WRITE (12,11)HEAD		CONF0680 16
WRITE (9,12)DELTA,GTMAX,GTEST,F308		CONF0690 18
WRITE (12,12)DELTA,GTMAX,GTEST,F308		CONF0700 19
WRITE (9,13)NGT,IFGF,IPP,INTGF,NAVE,IPUN,IYP,IFLD		CONF0710 20
WRITE (12,13)NGT,IFGF,IPP,INTGF,NAVE,IPUN,IYP,IFLD		CONF0720
C		21

LIN=0		
DELA=0.0		
C		
IF(DELTA)99,80,99		
80 IF(F3DB)81,82,81		
81 DELTA=66.66667/F3DB		
GO TO 83		
82 DELTA=0.0001		
83 WRITE(9,17)DELTA		29
WRITE(12,17)DELTA		30
99 IF(IFGF)106,106,101	CONF077C	
106 WRITE (9,16)	CONF078C	33
WRITE (12,16)	CONF079C	34
GO TO 100		
C	CONF086C	
C	CONF087C	
BEGINNING OF LOOP TO READ AND FOLD MULTIPLE INPUT G	CONF088C	
C	CONF089C	
101 IF(NGT)102,102,103	CONF090C	
102 NGT=1		
103 IDIM=999		
DO 2000 JNC=1,NGT	CONF091C	
C	CONF092C	
C	CONF093C	
READ GF	CONF094C	
C	CONF099C	
CALL RDOT(TEM,GF,IFGF,NGF,LABEL,IDIM,JOV)		44
IF(JOV.GT.0)WRITE(9,18)		46
IF(JOV.GT.0)WRITE(12,18)		
C	CONF101C	
C	CONF102C	
IF T NOT INPUT, TRANSFER TABLES AND GET T	CONF103C	48
C	CONF104C	
GO TO (210,201,210,201),IFGF	CONF105C	
201 GF(1)=TEM(1)	CONF106C	
TEM(1)=0.0	CONF107C	
DO202I=2,NGF	CONF108C	
GF(I)=TEM(I)	CONF109C	
202 TEM(I)=TEM(I-1)+DELTA	CONF110C	
C		
C		
LEFT DRESS TABLE		
C		
210 T1=TEM(1)		
DO 205 I=1,NGF		
205 TEM(I)=TEM(I)-T1		
C		
WRITE (12,4)	CONF111C	66
CALL GRAPH(1,NGF,TEM,GF,42,1)	CONF112C	67
CALL LINCNT(61)	CONF113C	69
WRITE (12,33)HEAD(1),JNG	CONF114C	70
CALL LINCNT(61)	CONF115C	71
WRITE (12,48)LABEL	CONF116C	
C	CONF117C	
C	CONF118C	
DIFFERENTIATE GF	CONF119C	72
C	CONF120C	
IF(INTGF)1213,1213,211	CONF121C	
211 NGF=NGF-1	CONF122C	
DO212I=1,NGF	CONF123C	
GF(I)=(GF(I+1)-GF(I))/(TEM(I+1)-TEM(I))	CONF124C	
212 TEM(I)=(TEM(I)+TEM(I+1))/2.0	CONF125C	
C	CONF126C	
C	CONF127C	
SMOOTH GF		
C		
1213 IF(NGF.LT.7.CR.NAVE.LE.0)GO TO 213		
NGF=NGF-2		
DO 260 I=1,NGF		
GF(I)=.25*GF(I)+.5*GF(I+1)+.25*GF(I+2)		
260 TEM(I)=.25*TEM(I)+.5*TEM(I+1)+.25*TEM(I+2)		
C		
C		
NORMALIZE G(T) TO AN AREA OF 1		
C		
213 SUM=0.0		
DO 214 I=2,NGF		

214	SUM=SUM+(GF(I)+GF(I-1))*0.5*(TEM(I)-TEM(I-1))		
DO 215	I=1,NGF		
215	GF(I)=GF(I)/SUM		
C		CONF132C	
C	GET TMX AND DELTA	CONF133C	
C		CONF134C	
	IF(GTMAX)221,222,221		
221	IF(GTMAX-TEM(NGF))223,223,222	CONF136C	
222	TMX=TEM(NGF)	CONF137C	
	GO TO 224	CONF138C	
223	TMX=GTMAX	CONF139C	
224	IF(((TMX-TEM(1))/DELTA)-999.00)1223,1223,1224	CONF140C	
1224	DELTA=(TMX-TEM(1))/999.0	CONF141C	
	WRITE (9,15)DELTA	CONF142C	130
	WRITE(12,4)		131
	WRITE (12,15)DELTA	CONF143C	132
	IF(JNG-1)1225,1225,100	CONF144C	
1223	IF(NGT-1)1231,1231,1225		
C		CONF146C	
C	GET EQUALLY SPACED GF	CONF147C	
C		CONF148C	
1225	CALL EQSPC(TEM,GF,TEM2,DELTA,TMX,NGF,LIN)		137
DO 261	I=1,NGF		
261	GF(I)=TEM2(I)		
C		CONF150C	
C	FOLD G1 WITH G2 ETC	CONF151C	
C		CONF152C	
	IF(JNG-1)1231,2001,2002		
2001	NGMAX=NGF	CONF154C	
	GO TO 2005	CONF155C	
2002	IF(NGMAX-NGF)2100,2003,2102	CONF156C	
2100	MIN=NGMAX+1	CONF157C	
	DO 2101 I=MIN,NGF	CONF158C	
2101	G(I)=0.0	CONF159C	
	GO TO 2104	CONF160C	
2102	MIN=NGF+1	CONF161C	
	DO 2103 I=MIN,NGMAX	CONF162C	
2103	GF(I)=0.0	CONF163C	
2104	NGF=MAX0(NGMAX,NGF)	CONF164C	
2003	CALL FOLD(G,GF,TEM2,NGF,DELTA)		167
	DO 262 I=1,NGF		
262	GF(I)=TEM2(I)		
2004	IF(JNG-NGT)2005,2000,2000	CONF166C	
2005	DO 2006 I=1,NGF	CONF167C	
2006	G(I)=GF(I)	CONF168C	
C		CONF169C	
2000	CONTINUE	CONF170C	
C		CONF171C	
C	TEM(1)=0.0	CONF172C	
C		CONF173C	
	DO 2007 I=2,NGF	CONF174C	
2007	TEM(I)=TEM(I-1)+DELTA	CONF175C	
	TMX=TEM(NGF)		
	IF(GTMAX.EQ.C.O.AND.NGT.LE.1)GO TO 1231		
	WRITE (12,4)	CONF176C	198
	CALL GRAPH(1,NGF,TEM,GF,42,1)	CONF177C	199
	CALL LINCNT(61)	CONF178C	201
	IF(NGT.GT.1)WRITE (12,34)HEAD(1)		203
	IF(NGT.LE.1)WRITE (12,29)HEAD(1)		
C		CONF180C	
C	END OF FOLD MULTIPLE G LOOP	CONF181C	
C		CONF182C	
C		CONF208C	
C	EXTEND G TABLE IF TEM(NGF)-TEM(1) IS LESS THAN 40. TAL	CONF209C	
C		CONF210C	205
1231	IF(IFLD.GE.1.OR.F3DB.LE.0.0)GO TO 1234		
	IF(TM.X.GE.(5331.2/F3DB))GO TO 1234		
	TEM(NGF+1)=5331.2/F3DB		
	TMX=TEM(NGF+1)		
	GF(NGF+1)=0.0	CONF215C	
	NGF=NGF+1	CONF216C	

1237	IF((TMX/DELTA).LT.999.0)GO TO 1234	
	DELTA=(TMX-TEM(1))/999.0	CONF2190
	WRITE (12,4)	CONF2200 221
	WRITE (9,15)DELTA	CONF2210 222
	WRITE (12,15)DELTA	CONF2220
C		CONF2310
C	STR INP G IN TEMP TAB	
C		CONF2330 223
1234	NGD=NGF	CONF2340
	DO 1227 I=1,NGD	
	TEM3(I)=TEM(I)	
1227	TEM4(I)=GF(I)	
C		CONF2360
C	SETUP FOR FOLD OPERATION	CONF2370
C		CONF2380
	IF(IFLD)100,230,227	CONF2390
227	CALL EQSPC(TEM,GF,TEM2,DELTA,TMX,NGF,LIN)	
	SUM=0.0	238
	DO 263 I=1,NGF	
263	SUM=SUM+TEM2(I)*DELTA	
	TEM3(I)=TEM(I)	
	DO 264 I=1,NGF	
	IF(I.GT.1)TEM3(I)=TEM3(I-1)+DELTA	
	G(I)=TEM2(I)/SUM	
264	G2(I)=G(I)	
	IF(IPUN.EQ.1)CALL PUNE12(TEM,G,NGF,HEAD)	262
	CONF - EFN SOURCE STATEMENT - IFN(S) -	
	GO TO 4300	
C		CONF2480
C		CONF2490
C	GET GMAX AND G1ST	CONF2500
C		CONF2510
230	GMAX=GF(1)	CONF2520
	DO232I=2,NGF	CONF2530
232	GMAX=AMAX1(GMAX,GF(I))	CONF2540
C		CONF2550
	IF(GTEST)235,235,236	CONF2560
235	SUM=0.0	CONF2570
	DO 1235 I=2,NGF	CONF2580
	ICT=I	CONF2590
	SUM=SUM+((GF(I)+GF(I-1))*0.5*(TEM(I)-TEM(I-1)))	CONF2600
	IF(SUM-.02)1235,1236,1236	CONF2610
1235	CONTINUE	CONF2620
1236	GTEST=GMAX/GF(ICT)	CONF2630
236	G1ST=GMAX/GTEST	CONF2650
C		CONF2640
	G1=G1ST	CONF2660
C		CONF2670
C	BEGINNING OF CONVERGENCE TEST LOOP	CONF2680
C		CONF2690
	DO 1409 IGTES=1,20	CONF2700
	G1ST=GMAX-(GMAX-G1)*(1.0-(FLOAT(IGTES-1)/20.0))	CONF2710
	GTEST=GMAX/G1ST	CONF2720
C		CONF2730
C	POSITION G1ST AND SHIFT TABLES SO. THAT GF(1)=G1ST	CONF2740
C		CONF2750
237	IF(G1ST-GF(1))240,250,242	CONF2760
240	J=NGF+2	CONF2770
	DO241I=1,NGF	CONF2780
	J=J-1	CONF2790
	GF(J)=GF(J-1)	CONF2800
241	TEM(J)=TEM(J-1)+DELTA	CONF2810
	GF(1)=0.0	CONF2820
	TEM(1)=TEM(2)-DELTA	CONF2830
C		CONF2840
242	DO243I=1,NGF	CONF2850
	ICT=I	CONF2860
	IF(GF(I)-G1ST)243,243,244	CONF2870
243	CONTINUE	CONF2880
C		CONF2890

244	SLOPE=(TEM(ICT)-TEM(ICT-1))/(GF(ICT)-GF(ICT-1))	CONF2900	
	TEM(1)=TEM(ICT)-SLOPE*(GF(ICT)-G1ST)	CONF2910	
	GF(1)=G1ST	CONF2920	
	NGF=NGF-ICT+2	CONF2930	
C		CONF2940	
	DO 245 I=2,NGF	CONF2950	
	J=I+ICT-2	CONF2960	
	GF(I)=GF(J)	CONF2970	
245	TEM(I)=TEM(J)	CONF2980	
C		CONF2990	
C	INTERPOLATE STARTING WITH G1ST	CONF3000	
C		CONF3010	
250	CALL EQSPC(TEM,GF,TEM2,DELTA,FMX,NGT,LIN)		
C		CONF3050	
C		CONF3060	
C	RE NORMALIZE	CONF3070	
C		CONF3080	332
	SUM=0.0		
	DO 271 I=1,NGT	CONF3100	
271	SUM=SUM+TEM2(I)*DELTA		
	DO 274 I=1,NGT		
274	G2(I)=TEM2(I)/SUM		
C		CONF3140	
C	CALCULATE GAUSSIAN	CONF3150	
C		CONF3160	
	FKDB=FKDD	CONF3170	
	CALL GAUSS(G,DELTA,FKDB,NGA,NGT,DELA)	CONF3180	345
	IREF=NGT/4		
C		CONF3200	
C		CONF3210	
C	UNFOLD GAUSSIAN WITH GF	CONF3220	
C		CONF3230	
307	TFST=TEM(1)	CONF3240	
	FLMA=NGT-IREF		
	TEM2(1)=G(1)/(G2(1)*DELTA)		
C		CONF3270	
	DO 402 J=2,NGT	CONF3280	
	SUM=0.0	CONF3290	
	K=J+1	CONF3300	
	MAX=J-1	CONF3310	
C		CONF3320	
	DO 401 I=1,MAX	CONF3330	
	K=K-1	CONF3340	
401	SUM=SUM+G2(K)*TEM2(I)	CONF3350	
C		CONF3360	
	TEM2(J)=(G(J)-SUM*DELTA)/(G2(1)*DELTA)		
C		CONF3380	
	IF(J-IREF)402,1402,1405	CONF3390	
1402	FMIN=TEM2(1)	CONF3400	
	FMAX=TEM2(1)	CONF3410	
C		CONF3420	
	DO 1403 K=1,IREF	CONF3430	
	FMIN=AMIN1(FMIN,TEM2(K))	CONF3440	
1403	FMAX=AMAX1(FMAX,TEM2(K))	CONF3450	
	GO TO 402	CONF3460	
C		CONF3470	
1405	IF(TEM2(J)-FMAX*(1.05-(FLOAT(J-IREF)/FLMA)))1406,402,1409	CONF3480	
1406	IF(TEM2(J)-FMIN*(1.05-(FLOAT(J-IREF)/FLMA)))1409,402,402	CONF3490	
C		CONF3500	
402	CONTINUE	CONF3510	
	GO TO 407	CONF3520	
1409	CONTINUE	CONF3530	
C		CONF3540	
C		CONF3550	
407	TEM(1)=TFST	CONF3560	
	NGF=NGT		
	DO 410 I=2,NGF	CONF3570	
410	TEM(I)=TEM(I-1)+DELTA	CONF3580	
	WRITE (12,4)	CONF3590	390
	CALL GRAPH(1,NGF,TEM,G2,42,1)	CONF3600	391
	CALL LINCNT(59)		393

WRITE (12,35)HEAD(1),GTEST	CONF3620	394
WRITE (12,36)	CONF3630	
C	CONF3640	395
408 DO 403 I=1,NGF	CONF3650	
403 G2(I)=TEM2(I)	CONF3660	
DELB=0.0	CONF3700	
C	CONF3750	
C GET G(I) FROM TEMP TABLE AND FOLD WITH INVERSE		
C	CONF3770	
1500 DO 1426 I=1,NGD		
TEM(I)=TEM3(I)		
1426 G(I)=TEM4(I)		
CALL EQSPC(TEM,G,TEM2,DELTA,TMX,NGD,LIN)		
C	CONF3800	416
SUM=0.0		
DO 426 I=1,NGF		
426 SUM=SUM+TEM2(I)*DELTA		
DO 427 I=1,NGF		
427 G(I)=TEM2(I)/SUM		
C		
CALL FOLD(G2,G,TEM2,NGF,DELTA)		433
DO 428 I=1,NGF		
428 G(I)=TEM2(I)		
C	CONF3840	
TEM3(1)=TEM(1)		
DO 4404 I=2,NGF		
TEM(I)=TEM(I-1)+DELTA	CONF3850	
4404 TEM3(I)=TEM(I)		
WRITE (12,4)	CONF3870	452
CALL GRAPH(1,NGF,TEM,G,42,1)	CONF3880	453
CALL LINCNT(61)	CONF3890	455
WRITE (12,51)HEAD(1)	CONF3900	
C	CONF3910	
C GET FREQ. RESP. AND PHASE OF UNFOLDED IMPULSE RESPONSE	CONF3920	
C	CONF3930	456
4300 NC0E=0		
FMIN=.01*F3DB		
FMAX=9.0*F3DB		
IF(F3DB.EQ.0.0)FMIN=250.0/(TEM3(NGF)-TEM3(1))		
IF(F3DB.EQ.0.0)FMAX=250.0/DELTA		
NLP=1		
CALL FOUR(TEM3,G,NGF,GF,TEM,TEM2,NC0E,TD,FMIN,FMAX,NLP)		465
DO 4415 I=1,NC0E		
4415 GF(I)=ALOG10(GF(I))	CONF4000	
C		472
WRITE (12,4)		475
CALL GRAPH(3,NC0E,GF,TEM,42,1)	CONF4150	476
CALL LINCNT(30)	CONF4160	478
WRITE (12,53)	CONF4170	479
CALL LINCNT(60)	CONF4180	480
WRITE (12,54)HEAD(1)	CONF4190	481
CALL LINCNT(62)	CONF4200	482
IF(IFLD.GT.0)WRITE (12,30)		484
IF(IFLD.EQ.0)WRITE (12,55)		486
WRITE (12,4)	CONF4220	487
CALL GRAPH(3,NC0E,GF,TEM2,42,1)	CONF4230	488
CALL LINCNT(30)	CONF4240	490
WRITE (12,56)	CONF4250	491
CALL LINCNT(60)	CONF4260	492
WRITE (12,54)HEAD(1)	CONF4270	493
CALL LINCNT(62)	CONF4280	494
IF(TD.NE.0.0)WRITE(12,57)		496
IF(TD.EQ.0.0)WRITE(12,58)		498
IF(IFLD.GT.0)GO TO 406		
C	CONF4300	
C GET INPUT PULSER G AND FOLD WITH INVERSE	CONF4310	
C	CONF4320	
IF(IPP)1520,1520,1501	CONF4330	
1501 CALL RDOT(TEM,GF,IPP,NIPP,LABEL,IDIM,JCV)		505
IF(JCV.GT.0)WRITE(9,18)		507
IF(JCV.GT.0)WRITE(12,18)		
C TRANSFER TABLES AND GET T IF T NOT INPUT	CCCCCCCCCCCCCCCCCCCC	509

```

      IF(IIPP.EQ.1.OR.IPP.EQ.3)GO TO 4419
      GF(1)=TEM(1)
      TEM(1)=0.0
      DO 4418 I=2,NIPP
      GF(I)=TEM(I)
4418 TEM(I)=TEM(I-1)+DELTA
      GO TO 4421
C      LEFT DRESS TABLE          CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
4419 TO=TEM(1)
      DO 4420 I=1,NIPP
4420 TEM(I)=TEM(I)-TO
C      DIFFERENTIATE IF INPUT IS STEP CCCCCCCCCCCCCCCCCCCCCCCCCC
4421 IF(INTGF)1504,1504,1502
1502 NIPP=NIPP-1
      DO 1503 I=1,NIPP
      GF(I)=(GF(I+1)-GF(I))/(TEM(I+1)-TEM(I))
1503 TEM(I)=(TEM(I)+TEM(I+1))/2.0
C
1504 IF(TEM(NIPP).GT.TMX)GO TO 1508
      NIPP=NIPP+1
      TEM(NIPP)=TMX+DELTA
      GF(NIPP)=0.0
1508 CALL EQSPC(TEM,GF,TEM2,DELTA,TMX,NIPP,LIN)
C      NORMALIZE INTEGRAL CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      SUM=TEM2(1)*DELTA
      DO 429 I=2,NIPP
      SUM=SUM+TEM2(I)*DELTA
429 TEM(I)=TEM(I-1)+DELTA
C
      DO 1506 I=1,NIPP
1506 GF(I)=TEM2(I)/SUM
C
1507 WRITE (12,4)
      CALL GRAPH(1,NGF,TEM,GF,42,1)
      CALL LINCNT(61)
      WRITE (12,42)HEAD(1)
C      FOLD WITH INVERSE CCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      CALL FOLD(GF,G2,TEM2,NGF,DELTA)
      DO 1510 I=1,NGF
1510 G2(I)=TEM2(I)
C
1520 TEM(1)=0.0
      DO 1523 I=2,NGF
1523 TEM(I)=TEM(I-1)+DELTA
C
C      PUNCH, PRINT AND PLOT INVERSE
C
      WRITE (12,4)
      CALL GRAPH(1,NGF,TEM,G2,42,1)
      CALL LINCNT(61)
      WRITE (12,38)HEAD(1)
      WRITE (12,4)
      IF(IPUN.NE.1)GO TO 406
405 WRITE (9,14)GTEST
      WRITE (12,14)GTEST
      WRITE (9,31)
      WRITE (12,31)
      WRITE (9,32)(TEM(I),G2(I),I=1,NGF)
      WRITE (12,32)(TEM(I),G2(I),I=1,NGF)
      CALL PUNE12(TEM,G2,NGF,HEAD)
406 IF(IYP.EQ.0)GO TO 100
C
C      FOLD G(T) WITH DATA
C
      CALL FIWF(DELTA,DELA,NGF,IPUN,IFLD)
C
      GO TO 100
C
      END

```

CONF4350
CONF4360
CONF4370
CONF4380
CONF4390
CONF4400

560

CONF4500
CONF4510
CONF4520
CONF4530
CONF4540
CONF4550

579
580
582
583
584

CONF4640
CONF4650
CONF4660
CONF4670
CONF4680
CONF4690
CONF4700
CONF4710
CONF4720
CONF4730

602
603
605
606
607

CONF4750
CONF4760
CONF4770
CONF4780
CONF4790
CONF4800
CONF4810

611
612
613
614
615
623
631

CONF4820
CONF4840

635

04/28/00

RDDT - EFN SOURCE STATEMENT - IFN(S) -

```

C SUBROUTINE RDDT(X,Y,IFGF,NPTS,LABEL,IDIM,JOV)
C SUBROUTINE RDDT, READS 6E12 OR 7E10 WITH OR WITHOUT X TABLE RDDT0020
C IF IFGF=1 OR 3 READS TABLE OF X, BLANK AND TABLE OF Y, BLANK
C IF IFGF=2 OR 4 READS SINGLE TABLE PLUS BLANK
C IF IFGF=1 OR 2 READS 6E12 OR 6F12
C IF IFGF=2 OR 4 READS 7E10 OR 7F10
C IDIM=DIMENSION SIZE OF X OR Y IN MAIN PROGRAM, INPUT TO CALL ARGU.
C JOV IS AN OUTPUT WHICH IS GT 0 IF MORE THAN IDIM POINTS HAVE
C BEEN READ. EXCESS POINTS ARE NOT STORED OTHER THAN IN THE Z LOC.
C
C DIMENSION X(1),Y(1),Z(7) RDDT0040
C
C 3 FORMAT(6E12.0,A6) RDDT0060
C 4 FORMAT(6E12.0) RDDT0070
C 5 FORMAT(7E10.0,2XA6) RDDT0080
C 6 FORMAT(7E10.0) RDDT0090
C RDDT0100
C RDDT0110
C
C J=0
C JOV=0
C JY=0
C JP=0
C IF(IFGF.LE.2)JDEL=6
C IF(IFGF.GE.3)JDEL=7
C
C READ TABLE OF X
C
C 20 IF(JDEL.EQ.6)READ(10,3)(Z(I),I=1,JDEL),LAB 11
C IF(JDEL.EQ.7)READ(10,5)(Z(I),I=1,JDEL),LAB 20
C IF(J.EQ.0)LABEL=LAB
C
C SUM=0.0
C DO 26 I=1,JDEL
C 26 SUM=SUM+ABS(Z(I))
C IF(SUM.EQ.0.0)GO TO 40
C
C IF(J+JDEL.GT.IDIM)JOV=JOV+JDEL
C IF(JOV.GT.J)GO TO 20
C
C DO 28 I=1,JDEL
C K=I+J
C 28 X(K)=Z(I)
C J=J+JDEL
C JOV=J
C JP=J
C GO TO 20
C
C 40 IF(J.EQ.0.AND.JOV.EQ.J)GO TO 60
C IF(J.EQ.0.AND.JOV.NE.J)GO TO 50
C 41 IF(X(J).NE.0.0)GO TO 50
C J=J-1
C GO TO 41
C
C READ TABLE OF Y
C
C 50 IF(IFGF.EQ.2.OR.IFGF.EQ.4)GO TO 60
C
C 52 IF(JDEL.EQ.6)READ(10,4)(Z(I),I=1,JDEL) 72
C IF(JDEL.EQ.7)READ(10,6)(Z(I),I=1,JDEL) 80
C
C DO 55 I=1,JDEL
C K=I+JY
C IF(K.LE.J)Y(K)=Z(I)
C IF(K.GT.JOV)GO TO 60
C 55 CONTINUE

```

JY=JY+JDEL
GO TO 52

C 60 NPTS=J
JOV=JOV-JP

C RETURN
END

RDDT058C
RDDT059C

EQSP - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

SUBROUTINE ECSPC(T,F,FE,DELTA,TMX,NGF,M)
DIMENSION T(1),F(1),FE(1)

C NGFE=(TMX-T(1))/DELTA+1.01
IF(TMX-T(1)-FLOAT(NGFE-1)*DELTA)252,253,253
252 NGFE=NGFE-1
253 S=T(1)+DELTA
N=2
DO 258 I=2,NGFE
254 IF(T(N)-S)255,256,257
255 N=N+1
GO TO 254
256 FE(I)=F(N)
GO TO 258
257 B=(F(N)-F(N-1))/(T(N)-T(N-1))
IF(M.GT.0)B=ALOG(F(N)/F(N-1))/(T(N)-T(N-1))
FE(I)=F(N-1)+B*(S-T(N-1))
IF(M.GT.0)FE(I)=EXP(ALOG(F(N-1))+B*(S-T(N-1)))
258 S=S+DELTA
C FE(1)=F(1)
NGF=NGFE
C RETURN
END

EQSP013C

EQSP015C
EQSP016C

EQSP018C
EQSP019C

EQSP021C

27

35

38

EQSP025C

EQSP026C
EQSP029C
EQSP030C
EQSP031C

FOLD - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

SUBROUTINE FOLD(G,F,T2,NGF,DT)
DIMENSION G(1),F(1),T2(1)

C DO 10 J=1,NGF
SUM=0.0
K=J+1
C DO 5 JK=1,J
K=K-1
5 SUM=SUM+G(K)*F(JK)
C 10 T2(J)=SUM*DT
C RETURN
C END

FOLD011C
FOLD012C
FOLD013C
FOLD014C
FOLD015C
FOLD016C
FOLD017C
FOLD019C
FOLD021C
FOLD024C
FOLD025C
FOLD026C

04/28/00

GAUSS - EFN SOURCE STATEMENT - IFN(S) -

```

SUBROUTINE GAUSS(G,DELTA,FDB,NGA,NGF,DELY)
DIMENSION G(1)
C
N=NGF
DELY=0.0
NGA=NGF/10
IF(FDB)300,300,301
300 IF(NGF.EQ.0)NGF=2
G(1)=1.0/DELTA
DO 1300 I=2,NGF
1300 G(I)=0.0
RETURN
C
301 TAU=132.28/FDB
302 A=1.0/(TAU*2.5066283)
FP=A
UD=1.0/DELTA
B=1.0/(UD*TAU)
IF(N.EQ.0)NGF=6.*TAU/DELTA+1.01
C
DO 303 I=2,NGF
ICTR=I
G(I)=A*EXP(-.5*((DELTA*FLOAT(I)/TAU)-B)**2)
IF(G(I)-(A/100.0))304,303,303
303 CONTINUE
304 FP=G(ICTR)
DELY=DELTA*FLOAT(ICTR-1)
UD=1.0/(DELTA+TAU*SQRT(2.0*ALOG(1.0/(FP*TAU*2.5066283))))
B=1.0/(UD*TAU)
IF(N.EQ.0)NGF=2*ICTR-1
C
DO 305 I=1,NGF
ARGU=0.5*((DELTA*FLOAT(I)/TAU)-B)**2
IF(ARGU.GT.69.0)G(I)=0.0
IF(ARGU.LE.69.0)G(I)=A*EXP(-ARGU)
305 CONTINUE
C
NGA=10.0*TAU/DELTA
C
SUM=0.0
DO 306 I=1,NGF
306 SUM=SUM+G(I)*DELTA
DO 307 I=1,NGF
307 G(I)=G(I)/SUM
C
RETURN
END

```

GAUSS0030
GAUSS0110
GAUSS0120
GAUSS0130
GAUSS0140
GAUSS0150
GAUSS0160
GAUSS0170
GAUSS0180
GAUSS0190
GAUSS0210
GAUSS0220
GAUSS0240
GAUSS0250
GAUSS0260
GAUSS0270
GAUSS0280 23
GAUSS0290
GAUSS0300
GAUSS0310
GAUSS0320
GAUSS0330 32 33
GAUSS0340
GAUSS0350
GAUSS0370
43
GAUSS0420
GAUSS0430
GAUSS0440
GAUSS0460

04/28/00

FOUR - EFN SOURCE STATEMENT - IFN(S) -

```

SUBROUTINE FOUR(T,G,NPT,FR,DB,PHAS,NFR,DLY,FMIN,FMAX,NLP)
GET FREQ. RESP. AND PHASE
DIMENSION DB(1),PHAS(1),FR(1),G(1),T(1)
C
T=TIME IN NS
C
G= G TABLE TO BE EVALUATED
C
DB= ACCEPTANCE IN DB
C
PHAS= PHASE ANGLE IN DEG.
C
FR= FREQ. IN MC.
C
IF NLP IS NON ZERO, ATTEMPTS LINEAR PHASE REMOVAL

```



```

C
C      COMPUTE TABLE OF FR IF NOT INPUT
C
      IF(NFR.GT.0)GO TO 90
      DO 80 I=3,102
80    FR(I)=FMIN*10.0**((FLOAT(I-3)*ALOG10(FMAX/FMIN)/99.0))
      NFR=102
      GO TO 100
C
      90 NFR=NFR+2
      DO 95 I=3,NFR
      J=NFR+3-I
      95 FR(J)=FR(J-2)
C
      100 FR(1)=.01*FMIN
      FR(2)=.1*FMIN
C
C      EVALUATE FOURIER INTEGRAL
C
      DO 125 I=1,NFR
C
      A=0.0
      B=0.0
      PF=.001*6.2831853*FR(I)
      C01=COS(PF*T(1))
      SN1=SIN(PF*T(1))
C
      DO 110 J=2,NPT
      DTH=.5*(T(J)-T(J-1))
      C02=COS(PF*T(J))
      SN2=SIN(PF*T(J))
C
      A=A+DTH*(G(J-1)*C01+G(J)*C02)
      B=B+DTH*(G(J-1)*SN1+G(J)*SN2)
C
      C01=C02
      110 SN1=SN2
C
      B=-B
      C=SQRT(A**2+B**2)
      DB(I)=-20.0*ALOG10(1.0/C)
C
C      EVALUATE PHASE
C
      PHA=57.295779*ATAN(ABS(B/A))
      IF(A)115,114,114
      114 IF(B)118,125,125
      115 IF(B)117,116,116
      116 PHA=180.0-PHA
      GO TO 125
      117 PHA=PHA+180.0
      GO TO 125
      118 PHA=360.0-PHA
C
      125 PHAS(I)=PHA
C
      DO 134 I=1,NFR
      J=NFR+2-I
      134 PHAS(J)=PHAS(J-1)
      PHAS(1)=0.0
C
      NFRT=NFR+1
      DO 140 I=2,NFRT
C
      DPHA=PHAS(I)-PHAS(I-1)
      DPHA=DPHA-360.0*AIN(T(DPHA/360.0))
      IF(ABS(DPHA)-180.0)139,139,135
      135 DPHA=DPHA-SIGN(360.0,DPHA)
      139 PHAS(I)=PHAS(I-1)+DPHA
C
      140 CONTINUE

```

```

C
DO 141 I=1,NFR
141 PHAS(I)=PHAS(I+1)
C
CCCCCCCCC REMOVE LINEAR PHASE CCCCCCCCCC
C IF NON LIN COMP. AT .1*FMIN .LE. 1. PERCENT OF
C TOTAL PHASE SHIFT, TEST CONSIDERS PHASE SHIFT
C AT .01*FMIN TO BE LINEAR
C
DLY=0.0
IF(NLP.EQ.0)GO TO 137
IF(.01.LT.(ABS(PHAS(2)-(PHAS(1)*FR(2)/FR(1))/PHAS(2)))GO TO 137
DLY=2.777778*PHAS(1)/FR(1)
C
PHREF=PHAS(1)
DO 136 I=1,NFR
136 PHAS(I)=PHAS(I)-PHREF*FR(I)/FR(1)
C
137 NFR=NFR-2
DO 138 I=1,NFR
FR(I)=FR(I+2)
138 PHAS(I)=PHAS(I+2)
C
DO 150 I=2,NFR
DPHA=PHAS(I)-PHAS(I-1)
DPHA=DPHA-360.0*AIN(T(DPHA/360.0)
IF(ABS(DPHA)-180.0)149,149,145
145 DPHA=DPHA-SIGN(360.0,DPHA)
149 PHAS(I)=PHAS(I-1)+CPHA
C
150 CONTINUE
C
160 RETURN
END

```

FIWF - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

```

SUBROUTINE FIWF(DT,DELY,NGF,IPUN,IFLD)
COMMON TEM(3000),TEM2(3000),G(3000),G2(1000),GF(1000),HEAD(12)
C
1 FORMAT(3E10.0)
2 FORMAT(3I5)
C
3 FORMAT(1H1,3X5HXMIN=1PE12.4,2X5HXMAX=E12.4,2X6HYZERO=E12.4,/1H )
33 FORMAT(1H ,3X9HDATA FMT=15,2X10HFRONT FIT=15,2X2HM=15,/1H )
C
4 FORMAT(1H1)
C
5 FORMAT(1H ,10X26HCATA EXCEEDS DIM STATEMENT)
6 FORMAT(14X,1A6,35X13HINPUT X AND Y)
7 FORMAT(1H1,20X44HEQUALLY SPACED X AND Y,EXCEEDS DIM STATEMENT)
8 FORMAT(14X,1A6,40X18HFRONT FITTED INPUT)
13 FORMAT(1H ,3X1P2E12.4)
14 FORMAT(14X,1A6,43X10HUNFOLDED Y)
15 FORMAT(1H1,7X1HX,10X10HUNFOLDED Y,/1H )
16 FORMAT(14X,1A6,43X8HFOLDED Y)
17 FORMAT(1H1,7X1HX,10X8HFOLDED Y,/1H )
C
C READ AND WRITE CONTROL CARDS
C
100 READ (10,1)TMIN,TMAX,YZERO
READ (10,2)ICF,IFFT,M
C
WRITE(12,4)
WRITE (9,3)TMIN,TMAX,YZERO
WRITE (12,3)TMIN,TMAX,YZERO

```

DATA0120
DATA013C
DATA014G
DATA0150
DATA016G
DATA017C
DATA0180
DATA019C
DATA020C

DATA0220
DATA023C
DATA024C
DATA0290
DATA0300

DATA032C
DATA033C
DATA034G
DATA035C
DATA036C
DATA037C
DATA038C
DATA0390

1
2
6
7
8

WRITE (9,33)IDF,IFFT,M	DATA0400	9
WRITE (12,33)IDF,IFFT,M	DATA0410	
C READ DATA	DATA0420	
C	DATA0430	10
IDIM=3199		
NF=0		
L=1		
JOP=0		
DO 215 K=1,151		
C		
IDIM=2999-NF		
CALL RDDT(TEM(L),G(L),IDF,NPC,LABEL,IDIM,JOV)		19
IF(NPC.EQ.0.AND.JOV.EQ.0)GO TO 201		
JOP=JOV		
NF=NF+NPC		
L=L+NPC		
C		
215 CONTINUE		
C	DATA0740	
201 IF(JOP.GT.0)WRITE(9,5)		32
IF(JOP.GT.0)WRITE(12,5)		
C		
C	DATA0490	
CCCCCCCC IF T NOT INPUT GET T	DATA0500	34
GO TO (220,202,220,202),IDF	DATA0510	
202 G(I)=TEM(I)	DATA0520	
TEM(I)=0.0	DATA0530	
DO 203 I=2,NF	DATA0540	
G(I)=TEM(I)	DATA0550	
203 TEM(I)=TEM(I-1)+DT	DATA0560	
GO TO 220	DATA0570	
C	DATA0580	
C	DATA0770	
SORT G WITH RESPECT TO TEM	DATA0780	
C	DATA0790	
220 CALL SORT(NF,TEM,G)	DATA0800	
C	DATA0810	
C	DATA0820	
AVERAGE POINTS WITH EQUAL TIMES, RESET COUNTERS AND TABLES	DATA0830	50
C	DATA0840	
IC=0	DATA0850	
DO 238 K=1,NF	DATA0860	
I=K+IC	DATA0870	
SUM=G(I)	DATA0880	
C	DATA0890	
IF(I-NF)232,231,239	DATA0900	
231 JC=JC+1	DATA0910	
GO TO 235	DATA0920	
C	DATA0930	
232 DO 234 J=I,NF	DATA0940	
JC=J	DATA0950	
IF(TEM(J+1)-TEM(J))235,233,235	DATA0960	
233 SUM=SUM+G(J+1)	DATA0970	
IC=IC+1	DATA0980	
234 CONTINUE	DATA0990	
C	DATA1000	
235 IJ=JC-IC	DATA1010	
G(IJ)=SUM/FLOAT(JC-I+1)	DATA1020	
TEM(IJ)=TEM(I)	DATA1030	
C	DATA1040	
238 CONTINUE	DATA1050	
239 NF=NF-IC	DATA1060	
C	DATA1070	
C	DATA1080	
ELIMINATE POINTS NOT BOUNDED BY INPUT TMIN AND TMAX	DATA1090	
IF(ABS(TMAX)+ABS(TMIN))260,260,241	DATA1100	
241 IMIN=1	DATA1110	
IF(TMIN)243,242,243	DATA1120	
242 TMIN=TEM(1)-DT	DATA1130	
243 IF(TMAX)245,244,245	DATA1140	
244 TMAX=TEM(NF)+DT	DATA1150	
C		

245	DO 249 I=2,NF	DATA116C	
	IMAX=I	DATA1170	
	IF(TEM(I)-TMIN)249,247,246	DATA118C	
246	IF(TEM(I-1)-TMIN)247,248,248	DATA119C	
247	IMIN=I	DATA1200	
	GO TO 249	DATA1210	
C		DATA1220	
248	IF(TEM(I)-TMAX)249,249,250	DATA1230	
249	CONTINUE	DATA1240	
	GO TO 251	DATA1250	
250	IMAX=I-1	DATA1260	
C		DATA1270	
251	J=0	DATA1280	
	DO 252 I=IMIN,IMAX	DATA129C	
	J=J+1	DATA1300	
	TEM(J)=TEM(I)	DATA131C	
252	G(J)=G(I)	DATA1320	
	NF=IMAX-IMIN+1	DATA1330	
C		DATA1340	
C	PLOT INPUT T AND F	DATA1350	
C		DATA1360	
260	WRITE (12,4)	DATA137C	124
	CALL GRAPH(1,NF,TEM,G,42,1)	DATA138C	125
	CALL LINCNT(61)	DATA139C	127
	WRITE (12,6)HEAD(1)	DATA140C	
C		DATA141C	
C	FRONT FIT F	DATA142C	
C		DATA1430	128
	TC=0.0	DATA144C	
	IF(1/FFT)261,270,261	DATA145C	
261	TC=DT*FLOAT(NCF)	DATA146C	
	DO 262 I=1,NF	DATA147C	
	IJ=NF-I+2	DATA148C	
	TEM(IJ)=TEM(IJ-1)+TC	DATA1490	
262	G(IJ)=G(IJ-1)	DATA150C	
	TEM(1)=TEM(2)-TC	DATA151C	
	G(1)=YZERO	DATA152C	
	NF=NF+1	DATA153C	
C		DATA154C	
C	GET EQUALLY SPACED T AND F	DATA155C	
C		DATA156C	
270	IF((TEM(NF)-TEM(1))/DT-2998.0)272,272,271	DATA1570	
271	WRITE (9,7)	DATA158C	145
	WRITE (12,7)	DATA1590	146
	RETURN	DATA1600	
C		DATA1610	
272	TMX=TEM(NF)	DATA1620	
	CALL EQSPC(TEM,G,TEM2,DT,TMX,NF,M)		149
1274	DO 273 I=2,NF	DATA1640	
	G(I)=TEM2(I)		
273	TEM(I)=TEM(I-1)+DT	DATA1650	
C		DATA1660	
C	PLOT FRONT FITTED T AND F	DATA1670	
C		DATA168C	
	IF(1/FFT)275,400,275		
275	WRITE (12,4)	DATA170C	162
	CALL GRAPH(1,NF,TEM,G,42,1)	DATA171C	163
	CALL LINCNT(61)	DATA172C	165
	WRITE (12,8)HEAD(1)	DATA173C	
C		DATA1740	
C	UNFOLD INPUT FUNCTION	DATA2670	
C		DATA2680	166
400	CALL SGFLD(TEM,G,NF,G2,NCF,DT)		168
	DO 401 I=1,NF	DATA270C	
401	TEM(I)=TEM(I)-TC-CELY		
C		DATA272C	
C	PLOT UNFOLDED INPUT	DATA2730	
C		DATA2740	
	WRITE (12,4)	DATA2750	177
	CALL GRAPH(1,NF,TEM,G,42,1)	DATA276C	178
	CALL LINCNT(61)	DATA277C	180
	IF(1/FLD.LT.1)WRITE(12,14)HEAD(1)		182

<pre> C IF(IFLD.GE.1)WRITE(12,16)HEAD(1) C PRINT UNFOLDED F C IF(IFLD.LT.1)WRITE(9,15) IF(IFLD.LT.1)WRITE(12,15) IF(IFLD.GE.1)WRITE(9,17) IF(IFLD.GE.1)WRITE(12,17) WRITE (9,13)(TEM(I),G(I),I=1,NF) WRITE (12,13)(TEM(I),G(I),I=1,NF) IF(IPUN.NE.4)GO TO 402 CALL PUNE12(TEM,G,NF,HEAD) C 402 RETURN END </pre>	<pre> DATA2790 DATA2800 DATA2810 184 186 188 190 192 DATA2840 193 DATA2850 201 212 DATA2860 DATA2880 </pre>
--	--

SGFL - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

<pre> C SUBROUTINE SGFLD(T,F,NPF,G,NG,DT) C DIMENSION T(1),F(1),G(1) C SUBROUTINE SHORT G MATRIX FOLD C DO 10 J=NG,NPF IST=J-NG+1 SUM=0.0 K=NG+1 C DO 5 JK=IST,J K=K-1 5 SUM=SUM+G(K)*F(JK) C T(IST)=T(J) / 10 F(IST)=SUM*DT C NPF=NPF-NG C RETURN END </pre>	<pre> SGFL0020 SGFL0040 SGFL0120 SGFL0130 SGFL0140 SGFL0150 SGFL0160 SGFL0170 SGFL0180 SGFL019 SGFL0200 SGFL022 SGFL0230 SGFL0240 SGFL0250 SGFL0260 SGFL0270 </pre>
--	---

PUNE12 - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

<pre> C SUBROUTINE PUNE12(X,Y,N,LABEL) C PUNCH CARDS 6E12.5 C PUNCH DECIMAL DATA C PUNCH X TABLE WITH LABEL AND SEQUENCE NUMBER C PUNCH Y TABLE WITH LABEL AND SEQUENCE NUMBER C OMIT Y TABLE IF X ARGU. ALSO APPEARS IN Y LOC. OF CALL SEQUENCE C EACH TABLE ENDS WITH A BLANK CARD C DIMENSION X(1),Y(1),Z(6) C SUM=0.0 DO 5 I=1,N 5 SUM=SUM+ABS(X(I)-Y(I)) C NSEQ=1 DO 18 L=1,2 DO 16 I=1,N,6 DO 10 J=1,6 K=I+J-1 Z(J)=X(K) </pre>	<pre> PUNE0030 PUNE0040 PUNE0050 PUNE0060 PUNE0070 PUNE0080 PUNE0090 PUNE0100 PUNE0120 PUNE0130 PUNE0140 PUNE0150 PUNE0160 PUNE0170 PUNE0180 PUNE0190 PUNE0200 PUNE0210 PUNE0220 </pre>
--	---

```

      IF(L-2)10,9,10
9     Z(J)=Y(K)
10    CONTINUE
      IF(K-N)14,14,11
11    K=MOD(N,6)+1
      DO 12 J=K,6
12    Z(J)=0.0
14    WRITE (11,20)Z,LABEL,NSEQ
16    NSEQ=NSEQ+1
      WRITE (11,22)LABEL,NSEQ
      NSEQ=NSEQ+1
      IF(SUM)18,19,18
18    CONTINUE
19    RETURN
C
20    FORMAT(1P6E12.5,A6,I2)
22    FORMAT(72X,A6,I2)
      END

```

```

PUNE0230
PUNE0240
PUNE0250
PUNE0260
PUNE0270
PUNE0280
PUNE0290
PUNE0300 35
PUNE0310
PUNE0320 39
PUNE0330
PUNE0340
PUNE0350
PUNE0360
PUNE0370
PUNE0380
PUNE0390
PUNE0400

```

SORT - EFN SOURCE STATEMENT - IFN(S) -

04/28/00

```

SUBROUTINE SORT(N,X,Y)
DIMENSION X(1), Y(1)
M=N
2  M=M/2
   IF(M.EQ.0) RETURN
   K=N-M
   J=1
4  I=J
6  L=I+M
   IF(X(I).LT.X(L)) GO TO 8
   A=X(I)
   X(I)=X(L)
   X(L)=A
   A=Y(I)
   Y(I)=Y(L)
   Y(L)=A
   I=I-M
   IF(I.GE.1) GO TO 6
8  J=J+1
   IF(J.GT.K) GO TO 2
   GO TO 4
END

```


SK110E UNFOLD FOLDED 50 MHZ GAUSS WITH SK110E
 DELTA= -0. GTMAX= 3.0000E 01 GTEST= -0.

F3DB= 4.2500E 02

NGT= 1 FMT G= 3 FMT INP PULSE= -0 INTGF= -0 NAVE= -0 IPUN= 4 YP= 3 FLD= -0

COMPUTED DELTA= 1.5686E-01

XMIN= -0. XMAX= -0. YZERO= -0.
 DATA FMT= 1 FRONT FIT= 1 M= -0

X	UNFOLDED Y
-1.5549E 00	1.3050E 04
-1.1901E 00	1.3125E-04
-1.0412E 00	1.3192E-04
-8.8434E-01	1.3260E-04
-7.2748E-01	1.3330E-04
-5.7061E-01	1.3405E-04
-4.1375E-01	1.3493E-04
-2.5689E-01	1.3604E-04
-1.0003E-01	1.3753E-04
5.6837E-02	1.3969E-04
2.1370E-01	1.4307E-04
3.7056E-01	1.4867E-04
5.2743E-01	1.5790E-04
6.8429E-01	1.7231E-04
8.4115E-01	1.9326E-04
9.9801E-01	2.2163E-04
1.1549E 00	2.5783E-04
1.3117E 00	3.0203E-04
1.4686E 00	3.5447E-04
1.6255E 00	4.1531E-04
1.7823E 00	4.8562E-04
1.9392E 00	5.6532E-04
2.0961E 00	6.5612E-04
2.2529E 00	7.5846E-04
2.4098E 00	8.7366E-04
2.5666E 00	1.0029E-03
2.7235E 00	1.1474E-03
2.8804E 00	1.3085E-03
3.0372E 00	1.4873E-03
3.1941E 00	1.6848E-03
3.3510E 00	1.9019E-03
3.5078E 00	2.1394E-03
3.6647E 00	2.3979E-03
3.8215E 00	2.6781E-03
3.9784E 00	2.9809E-03
4.1353E 00	3.3069E-03
4.2921E 00	3.6561E-03
4.4490E 00	4.0296E-03
4.6059E 00	4.4254E-03
4.7627E 00	4.8429E-03
4.9196E 00	5.2807E-03
5.0764E 00	5.7380E-03
5.2333E 00	6.2139E-03
5.3902E 00	6.7070E-03
5.5470E 00	7.2151E-03
5.7039E 00	7.7353E-03
5.8608E 00	8.2640E-03
6.0176E 00	8.7975E-03
6.1745E 00	9.3326E-03
6.3313E 00	9.8663E-03
6.4882E 00	1.0396E-02
6.6451E 00	1.0917E-02
6.8019E 00	1.1425E-02
6.9588E 00	1.1915E-02
7.1157E 00	1.2383E-02
7.2725E 00	1.2825E-02
7.4294E 00	1.3236E-02
7.5862E 00	1.3615E-02
7.7431E 00	1.3956E-02
7.9000E 00	1.4257E-02
8.0568E 00	1.4515E-02