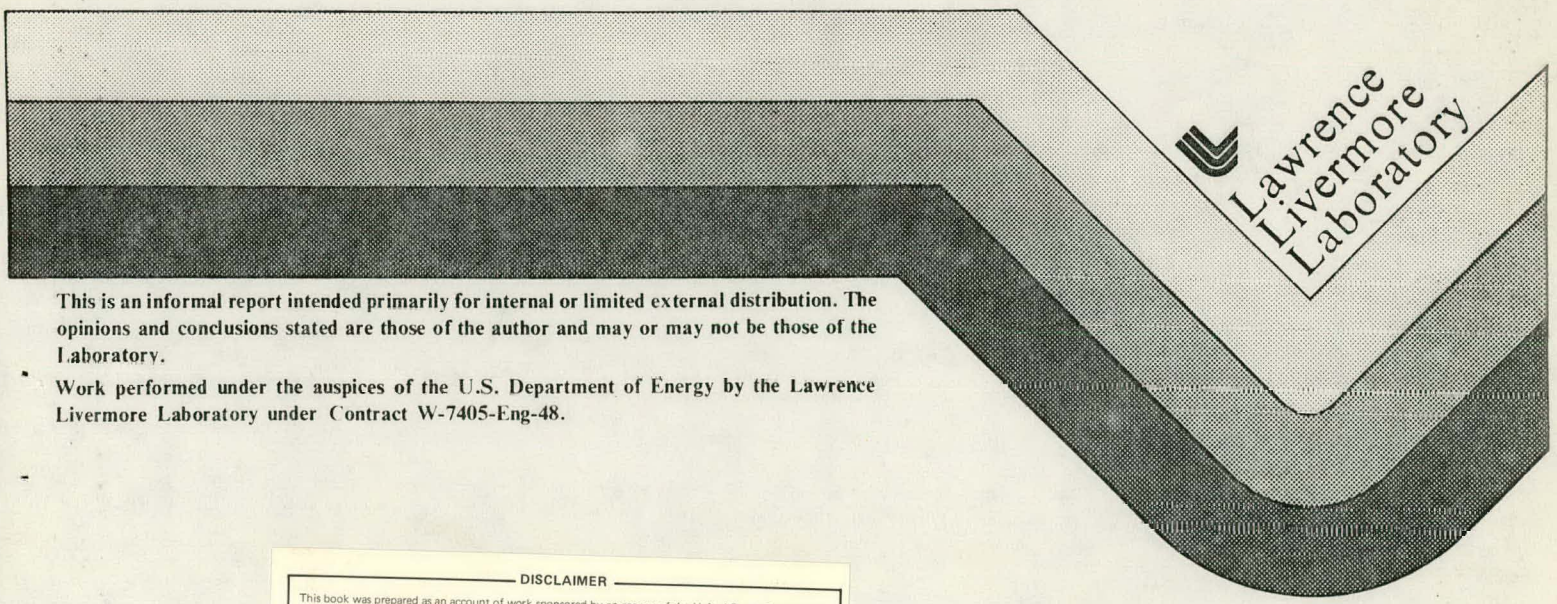
UCID--19268

DE82 006929

TRANSF Code User Manual

H. Joseph Weaver

November 1, 1981



Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

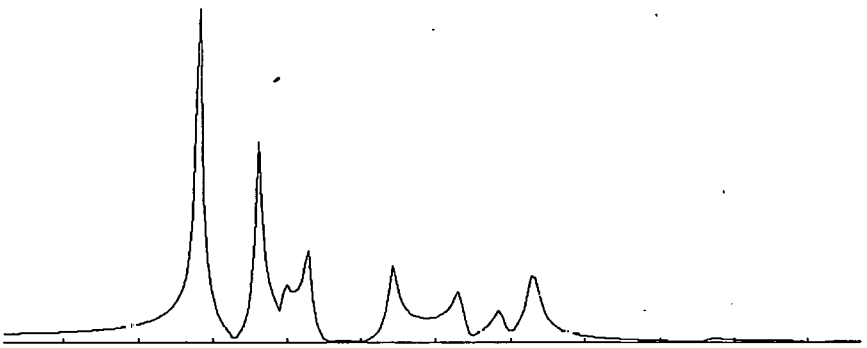
DISCLAIMER

UCID 19268

TRANSF

A Transfer Function Modal Analysis Code

User Manual



H. Joseph Weaver

Lawrence Livermore National Laboratory

University of California, Livermore Ca.

November 1, 1981

PREFACE

The TRANSF code is a semi-interactive FORTRAN IV program which is designed to caluclate the modal parameters of a (structural) system by performing a least squares parameter fit to measured transfer function data. The code is available at LLNL on both the 7600 and the Cray machines. The transfer function data to be fit is read into the code via a disk file. The primary mode of output is FR80 graphics, although, it is also possible to have some of your results written to either the TTY or to a disk file.

A copy of this report can be obtained on *any of the 7600 machines* by using TRIX AC as follows

```
 TRIX AC / .2 .3
.PRINT! NIPS D. .943794:TRANSF:JXTRANSF BOX Ann and yourid
.END
.ALL DONE
```

The code itself (TRANSF) can be obtained from my take directory on the 7600 machine as follows:

```
XPORT!  READ .943794:TRANSF:7600:TRANSF / t v
```

or on thc Cray Machine:

```
XPORT!  READ .943794:TRANSF:CRAY:TRANSF / t v
```

All questions and requests should be directed to

Joe Weaver 2-0310

TABLE OF CONTENTS

INTRODUCTION

In this manual we define a linear system as one that maps an input function to an output function. For example, a structural system could be one which converts a forcing function excitation at some location to a resulting response motion at another location. One way to characterize a linear system is via its transfer function which is, in effect, a measure of how well that system passes various single frequency complex exponential terms of the form

$$e(t) = e^{2\pi i w_0 t}.$$

This can be appreciated by noting that for a linear system e(t) is an *eigenfunction* which means that the resulting output [o(t)] will also be a single frequency complex exponential. Normally, however, the amplitude and phase of this output will be different from the input by a constant (with respect to t) factor of $H(w_0)$ [1]*, i.e.

$$o(t) = H(w_0)e(t).$$

As can be appreciated from this above equation, if we were to excite a system with several different frequency inputs one at a time and then plot the values of $H(w_0)$ (a complex number with amplitude and phase) vs w_0 we would obtain a graphical description of the transfer function. This technique is commonly called *Swept Sine Analysis* and has long been used to characterize a linear system. In recent years, however, this approach has become somewhat outdated due to the arrival of new technology and analysis techniques. Probably the most important advance being the development of the fast Fourier transform algorithm (FFT) by Cooley and Tukey [2] and the subsequent arrival of

---

[*] Numbers in brackets [] refer to the references listed at the end of this manual.

minicomputer spectrum analyzers that use a hardwired version of the FFT. These advances have now made it pratical to calculate the transfer function from the impulse response of the system. We define the *impulse response* of a system [denoted as h(t)] as the resulting output when the system is excited by a delta, or impulse, function. As it turns out [3-5], the transfer function of a system is just the Fourier transform of the impulse response function. Notationally, we represent this as

$$H(w) = \mathcal{F}[h(t)]$$

If a system can be (ideally) described by an N-th order differential equation then its impulse response function takes on the form of a sum of damped sinusoidal terms, i.e.

$$(1) \qquad h(t) = \sum_{k=1}^{N} A_k e^{-\sigma_k t} \sin(2\pi w_k t + \varphi_k).$$

Consequently, the transfer function (which is obtained by Fourier transforming equation 1) takes on the following form:

$$(2) \qquad H(w) = \sum_{k=1}^{N} \frac{B_k}{\sigma_k + 2\pi i(w - w_k)} \qquad w \geq 0^{*}.$$

$$\text{where:} \quad B_k = (A_k/2)e^{i\varphi_k}$$

In Figure 1 we show a single damped sinusoidal term as described by equation 1 (N=1) and in Figure 2 the corresponding Fourier transform as described by equation 2. In Figure 3 we again show H(w) only this time in its exponential form (amplitude and phase). Functions described by the basic form given in equation 2 are sometimes called *Lorentzian* functions. As can be seen in this equation, the denominator contains the imaginary unit element i (often crudely

---

[*] In our work we will only be interested in the transfer function for positive frequencies although the complete transform is also defined over negative values.

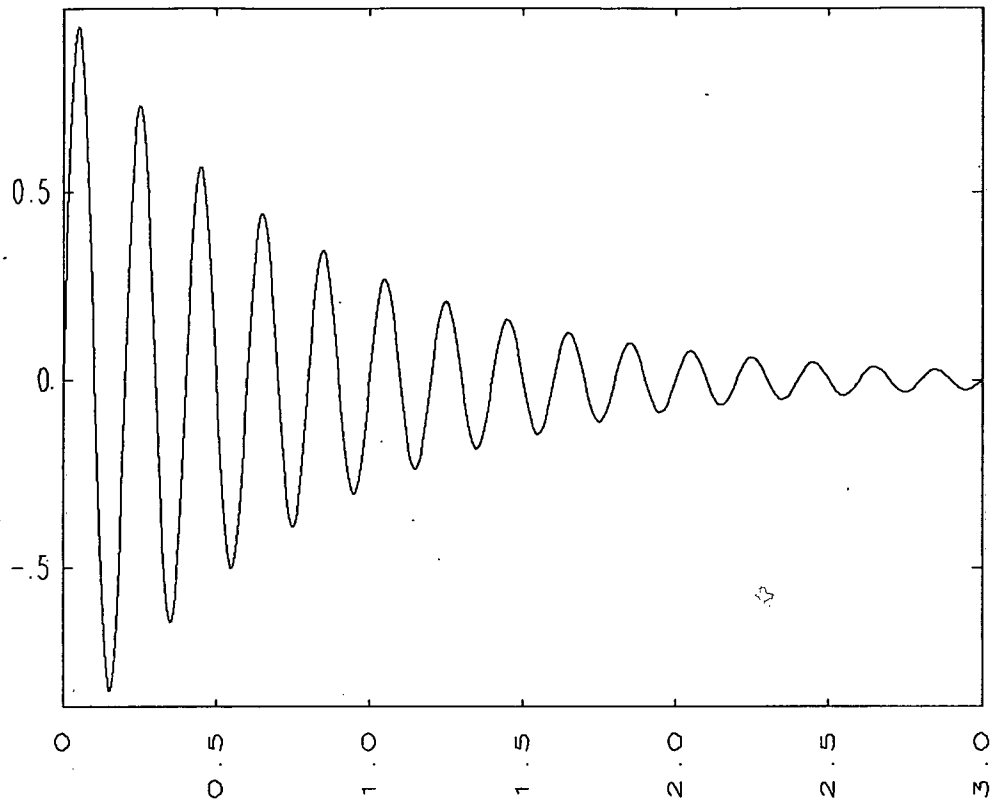


Figure 1. Damped Sinusoidal Function.

defined as the square root of negative one) and the numerator contains the complex number B_k. Consequently, Lorentzian functions are complex with a real and imaginary portion.

In equation 2 (as well as equation 1) there are certain sets of constants or parameters. Two of them are (σ_k and w_k) are real valued and the other (B_k) is complex. For most real systems these parameters turn out to have physical significance. Namely, w_k is the k-th natural or resonant frequency, σ_k is the k-th resonant damping and B_k is a measure of the relative strength and phase of this k-th component. The value of the frequency parameter w_k determines the peaks in both the imaginary portion (Figure 2) and the amplitude (Figure 3) of the transfer function H(w). On the other hand, the damping parameter σ_k determines the width. We will discuss this subject in greater detail in the next section.

To physically determine the transfer function of a system we must first excite the system with an impulse and then measure the resulting

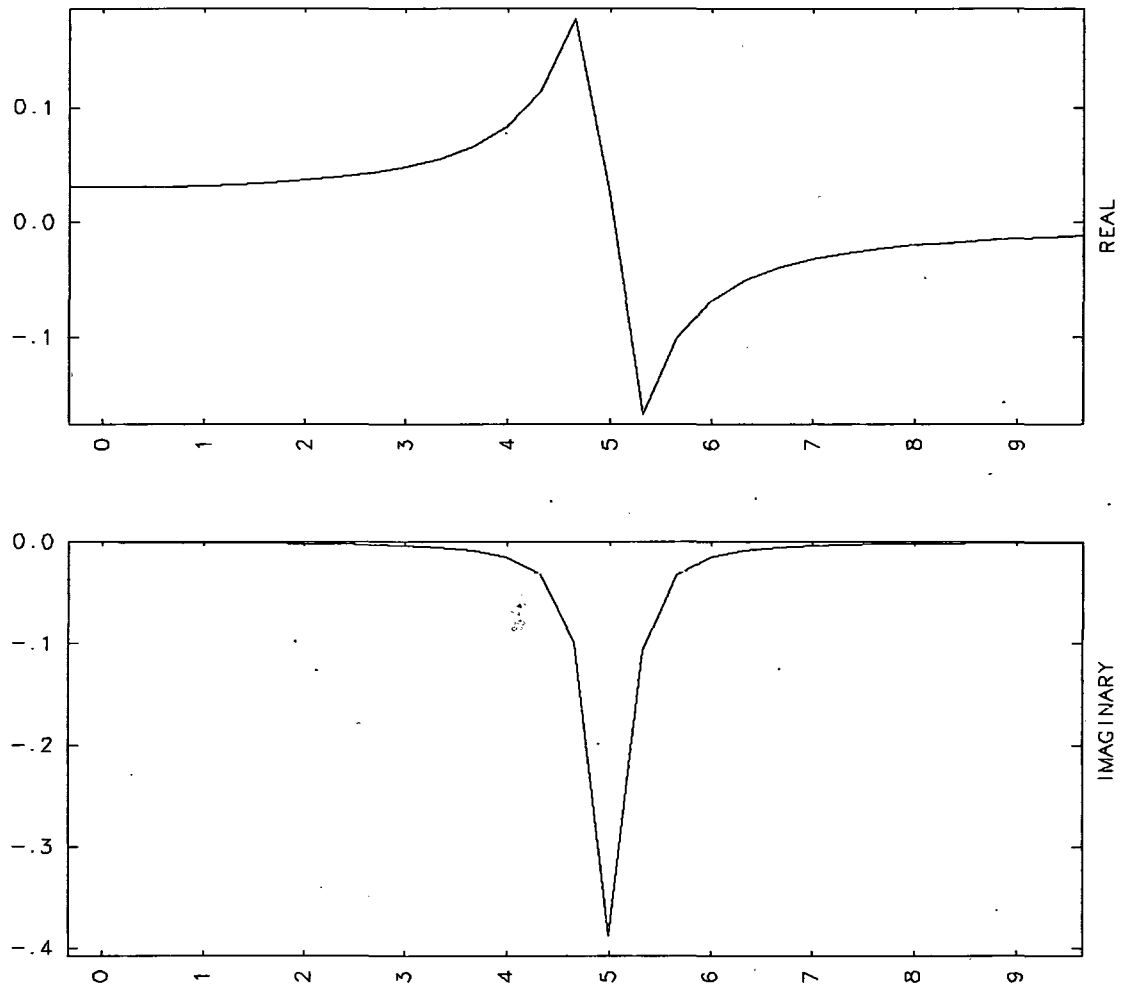


Figure 2. Lorentzian Function (Rectangular Form).

output, i.e. the impulse response. In real life, true impulse functions do not exist but instead only finite strength pulse functions of short duration. However, if the duration of the pulse is very small compared to the response characteristics of the system then for all practical purposes they may be considered impulses. For example, when testing structural systems, a hammer blow to the structure can almost always be considered an impulse excitation. Once the impulse response of the system is recorded, the transfer function

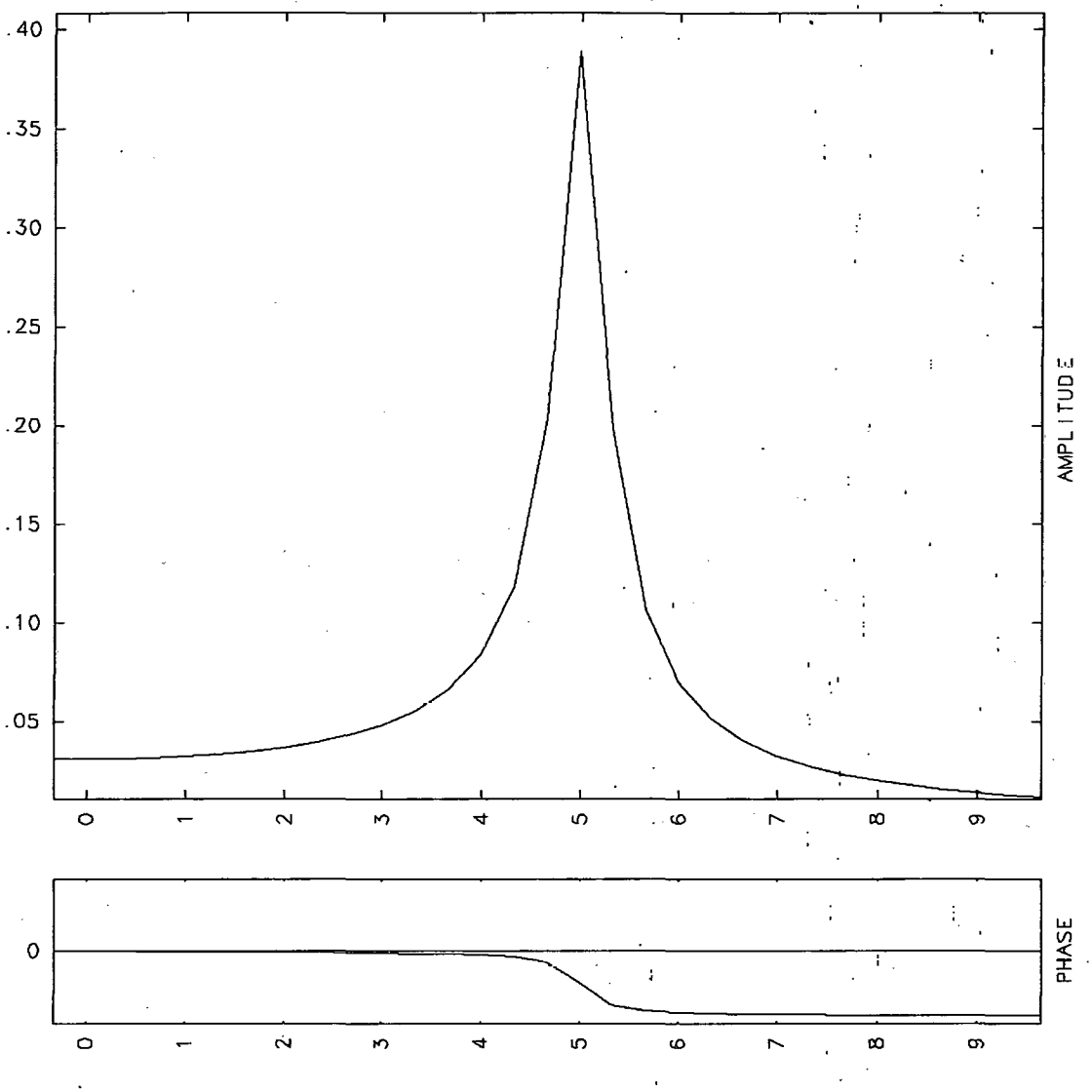


Figure 3. Lorentzian Function (Exponential Form).

can be obtained by Fourier transform techniques. These Fourier transforms are calculated on a digital computer using the efficient FFT algorithm.

We note here that on some occassions a satisfactorly short pulse (approximate impulse) excitation can not always be realized. When this is the case, additional Fourier preprocessing is required to obtain the transfer function. That is to say, the effect of the input

function must be deconvolved from the output function. Most Fourier analysis codes have this capability [6].

As far as the TRANSF code is concerned, the starting point is the measured and transformed data that represents the transfer function of the system. Thus, the problem at hand is to determine the system parameters B_k, σ_k and w_k from this measured transfer function. The TRANSF code accomplishes this by using a complex valued curve fitting scheme which we will describe in the next section. In the interest of instruction and clarity we will devote the remainder of this section to a discussion of another simple but less rewarding technique.

Again, for the sake of clarity, we will make the simplifying assumption that we only have one set of resonant parameters, In other words N=1 in equation 2, which results in

$$H(w) = \frac{B_0}{\sigma_0 + 2\pi i(w-w_0)} \qquad w \geq 0.$$

Taking the modulus squared of both sides of this above equation we find

$$(3) \qquad \|H(w)\|^2 = \frac{\|B_0\|^2}{\sigma_0^2 + 4\pi^2(w-w_0)^2} \qquad w \geq 0.$$

To determine the maximum of this function we take the derivative with respect to w and set the resulting expression equal to 0, i.e.

$$\frac{d\|H(w)\|^2}{dw} = \frac{8\pi^2\|B_0\|^2(w-w_0)}{(\sigma_0^2 + 4\pi^2(w-w_0)^2)^2} = 0.$$

From this we see that the maximum occures when $w=w_0$. Thus, to determine w_0 we examine the graph of equation 3 for its maximum value. If we now evaluate equation 3 at this value we obtain

$$(4) \qquad \|H(w_0)\|^2 = \frac{\|B_0\|^2}{\sigma_0^2}$$

Let us now designate w_c to be the frequency at which $\|H(w)\|^2$ is down to one half of its maximum value. i.e.

$$(5) \quad \|H(w_c)\|^2 = \frac{\|B_0\|^2}{2\sigma_0^2}$$

Substitution of $w=w_c$ into equation 3 and comparing the results to equation 5 we find

$$(6) \quad \sigma_0 = 2\pi(w_c - w_0).$$

Thus, to evaluate the damping parameter σ_0 we determine (from the graph of $\|H(w)\|^2$) the half maximum location w_c, subtract w_0 from it and multiply by 2π. Equation 4 can now be used to determine the value of $\|B_0\|^2$.

Instead of working with the graph of the modulus squared $\|H(w)\|^2$ we frequently use the graph of just the amplitude $\|H(w)\|$. In this case the half maxinum value of the modulus squared becomes the .707 value of $\|H(w)\|$. Also, rather than locating w_c and calculating $(w_c - w_0)$ we typically obtain the full width at this .707 value, i.e.

$$\Delta w_0 = 2|w_c - w_0|.$$

In this case the damping becomes

$$\sigma_0 = \pi \Delta w_0.$$

Finally, damping is commonly quoted in terms of a percent critical value ξ_0 as per the following formula

$$\sigma_0 = 2\pi w_0 \xi_0.$$

Thus,

$$(7) \qquad \xi_0 = \frac{\Delta w_0}{2w_0} \quad .$$

As an illustration, let's calculate the parameters of the data shown in Figure 3. As indicated in Figure 4 (dashed line) the resonant frequency is located at the peak value of $\|H(w)\|$ and is equal to 5 Hz. The width of the graph at the .707 level (arrows) turns out to be .4 Hz. Consequently, using equation 7 we determine ξ_0 to be 4 percent. Using this value of ξ_0 we find σ_0 to be equal to 1.26/sec. Finally, using this value in equation 4 with $\|H(5)\| = .4$ we obtain $\|B_0\| = .5$. It is interesting to note that when we use this technique we only obtain the amplitude, or modulus, of the complex number B_0 and not the phase.

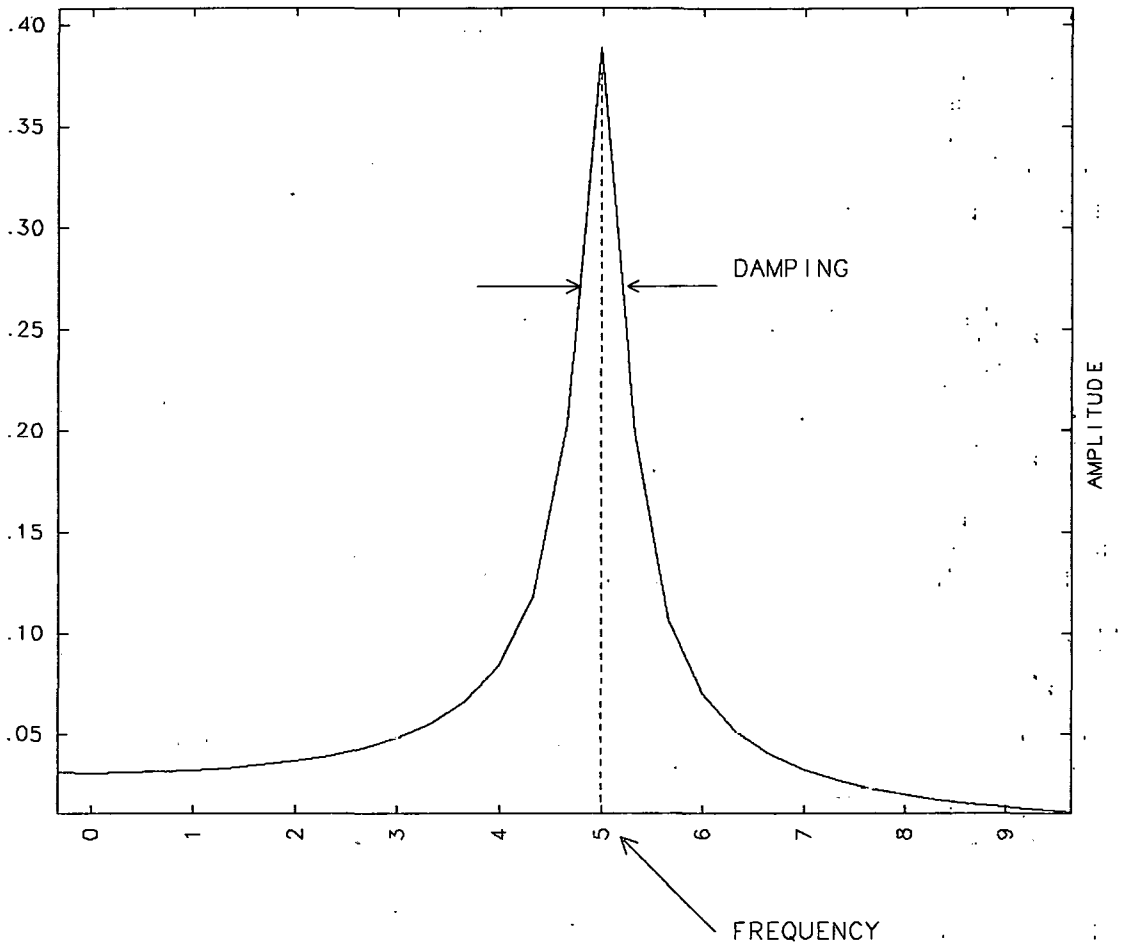


Figure 4. Example Calculation.

TRANSF CODE COMPUTATIONAL TECHNIQUE

In the previous section we illustrated a simple technique for extracting the dynamic (modal) parameters of a system from its measured transfer function. In that demonstration we tactifully assumed that the transfer function contained only one such mode. If this were not the case and there were two modes close together then the two graphs would overlap. This would have caused both a shift in the location of the peaks (frequency) and a spreading of the width of the curves (damping). Even when we only had one mode we could only determine the amplitude of the complex number B_0 using this simple technique. In this section we will describe the technique used by the TRANSF code to extract the modal parameters. Basically, this technique uses a curve fitting approach which eliminates the problem of closely spaced modes.

Before we discuss this technique we require the following results from Fourier analysis [3]:

$$\mathcal{F}[C_k e^{-\sigma_k t}\cos(2\pi w_k t)]^* = \frac{C_k/2}{\sigma_k + 2\pi i(w-w_k)} + \frac{C_k/2}{-\sigma_k + 2\pi i(w+w_k)} ,$$

$$\mathcal{F}[D_k e^{-\sigma_k t}\sin(2\pi w_k t)] = \frac{-iD_k/2}{\sigma_k + 2\pi i(w-w_k)} + \frac{iD_k/2}{\sigma_k + 2\pi i(w+w_k)} .$$

In our work we will only be concerned with frequencies greater than zero and we will make the assumption that for w>0 the second term in the above equations is negligable compared to the first. That is to say, for w>0 we have

$$\frac{1}{\sigma_k + 2\pi i(w+w_k)} \ll \frac{1}{\sigma_k + 2\pi i(w-w_k)} \qquad (w>0).$$

---

[*] Recall that $\mathcal{F}[f(t)]$ denotes the Fourier transform of $f(t)$.

Thus, the above equations become

$$(8) \quad \mathcal{F}[C_k e^{-\sigma_k t}\cos(2\pi w_k t)] = \frac{C_k/2}{\sigma_k + 2\pi i(w-w_k)},$$

$$(9) \quad \mathcal{F}[D_k e^{-\sigma_k t}\sin(2\pi w_k t)] = \frac{-iD_k/2}{\sigma_k + 2\pi i(w-w_k)}.$$

Now let us rewrite equation 1, which describes the impulse response of the system, and use the trigonometric identity: $\sin(\alpha+\beta) = \sin\beta\cos\alpha + \sin\alpha\cos\beta$, with $\alpha = 2\pi w_k t$ and $\beta = \varphi_k$. This results in

$$h(t) = \sum_{k=1}^{N} A_k\sin\varphi_k e^{-\sigma_k t}\cos(2\pi w_k t) + A_k\cos\varphi_k e^{-\sigma_k t}\sin(2\pi w_k t).$$

As noted in the previous section, the transfer function H(w) of the system is obtained by Fourier transforming the above equation. Using equations 8 and 9 (with $C_k=A_k\sin\varphi_k$ and $D_k=A_k\cos\varphi_k$) to accomplish this transform we find

$$H(w) = \sum_{k=1}^{N} \frac{(A_k/2)\sin\varphi_k}{\sigma_k + 2\pi i(w-w_k)} - \frac{(iA_k/2)\cos\varphi_k}{\sigma_k + 2\pi i(w-w_k)}$$

or

$$H(w) = \sum_{k=1}^{N} \frac{\hat{A}_k - \hat{B}_k}{\sigma_k + 2\pi i(w-w_k)},$$

where $\hat{A}_k = (A_k/2)\sin\varphi_k$ and $\hat{B}_k = (A_k/2)\cos\varphi_k$. Note that $\hat{A}_k$ and $\hat{B}_k$ are real numbers which are related to the real and imaginary portion of the complex number $(A_k/2)\exp[\varphi_k]$ as illustrated in Figure 6.

As our final step we will rationalize each term in the above equation by multipling the numerator and denominator by the complex conjugate of the denominator, i.e. $\sigma_k - 2\pi i(w-w_k)$. This move results in

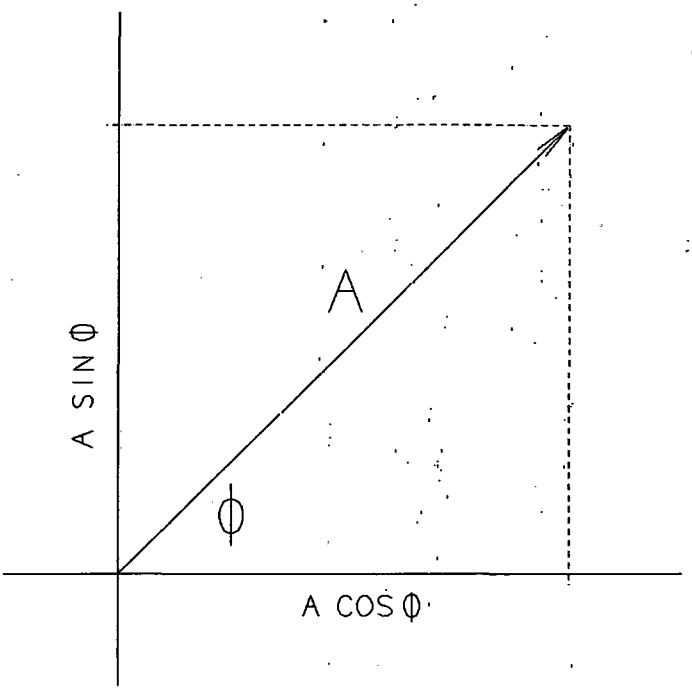


Figure 5. Representation of Complex Parameter $\hat{A}_k$.

$$(10) \quad H(w) = H_R(w) + iH_I(w),$$

where

$$(11) \quad H_R(w) = \sum_{k=1}^{N} \frac{\hat{A}_k \sigma_k - 2\pi \hat{B}_k (w-w_k)}{\sigma_k^2 + 4\pi^2 (w-w_k)^2}$$

$$(12) \quad H_I(w) = \sum_{k=1}^{N} \frac{-\hat{B}_k \sigma_k - 2\pi \hat{A}_k (w-w_k)}{\sigma_k^2 + 4\pi^2 (w-w_k)^2}$$

Thus, we now have expressions for both the real and imaginary portions of the transfer function. As can be seen in equations 11 and 12, for each term (k=1,..,N), the four parameters $\hat{A}_k$, $\hat{B}_k$, σ_k, and w_k can be used to completely describe the real or the imaginary portion. The approach taken by the TRANSF code is to use a least squares fit algorithm [7] to find the best set of these parameters so that the formulas given by equations 11 and 12 will best represent the measured

data. In other words, if our assumptions are satisfied and the system is reasonably linear then the transfer function should be described by a form of equations 10–12 when the values of the parameters are chosen correctly.

The measured transfer functions that are read into TRANSF are complex and, as we have already indicated, the resonant parameters can be obtained by fitting either the real or the imaginary portion of the data. Consequently, there is some redundancy in the data. TRANSF takes advantage of this fact in order to give a visual measure of how good a fit it produces. That is, the code first determines the parameters by fitting the real portion of the data to equation 11. The results are then calculated and plotted against the actual data. It next uses these same parameters in equation 12 to calculate and plot the projected imaginary fit against the actual imaginary data. This is illustrated in Figure 6 in which the real and imaginary parts are shown with a solid line and the fitted results as a dashed line. We emphasize here that up to this point the code has only analyzed the real portion of the data to obtain its results.

Next the procedure is reversed and the code works only with the imaginary portion of the data and obtains the parameters using equation 12. It then uses these parameters to project what the real portion should look like. This is illustrated in Figure 7 with the same dashed and solid line convention used in Figure 6. Finally, the code averages the two results and uses these averaged parameters to plot the calculated transfer functions (real and imaginary portions) against the actual data. This is illustrated in Figure 8.

We point out here that although the measured data in these figures actually contained 4 resonant modes, the code was deliberately restrained to find only three. This was done for the sake of demonstration; for if this were not the case then the dashed lines would lie essentially on top of the actual data and thus, would be invisible.

Currently the code uses a curve fitting routine (coded by Hank Finn) which is based on a method proposed by Bevington [7]. This algorithm is placed in a subroutine called *CURFIT* and is periodically updated or replaced as better ones come along.

In addition to the formulas given in equations 11 and 12, the code also adds a second degree polynomial of the form

$$Aw^2 + Bw + C.$$

This helps to compensate for any offset in the transfer function data that could otherwise bias the results.

Under contract to the United States Nuclear Regulatory Commission (NRC) this code was analytically qualified for use as a potential testing tool for nuclear power plant structures [8].

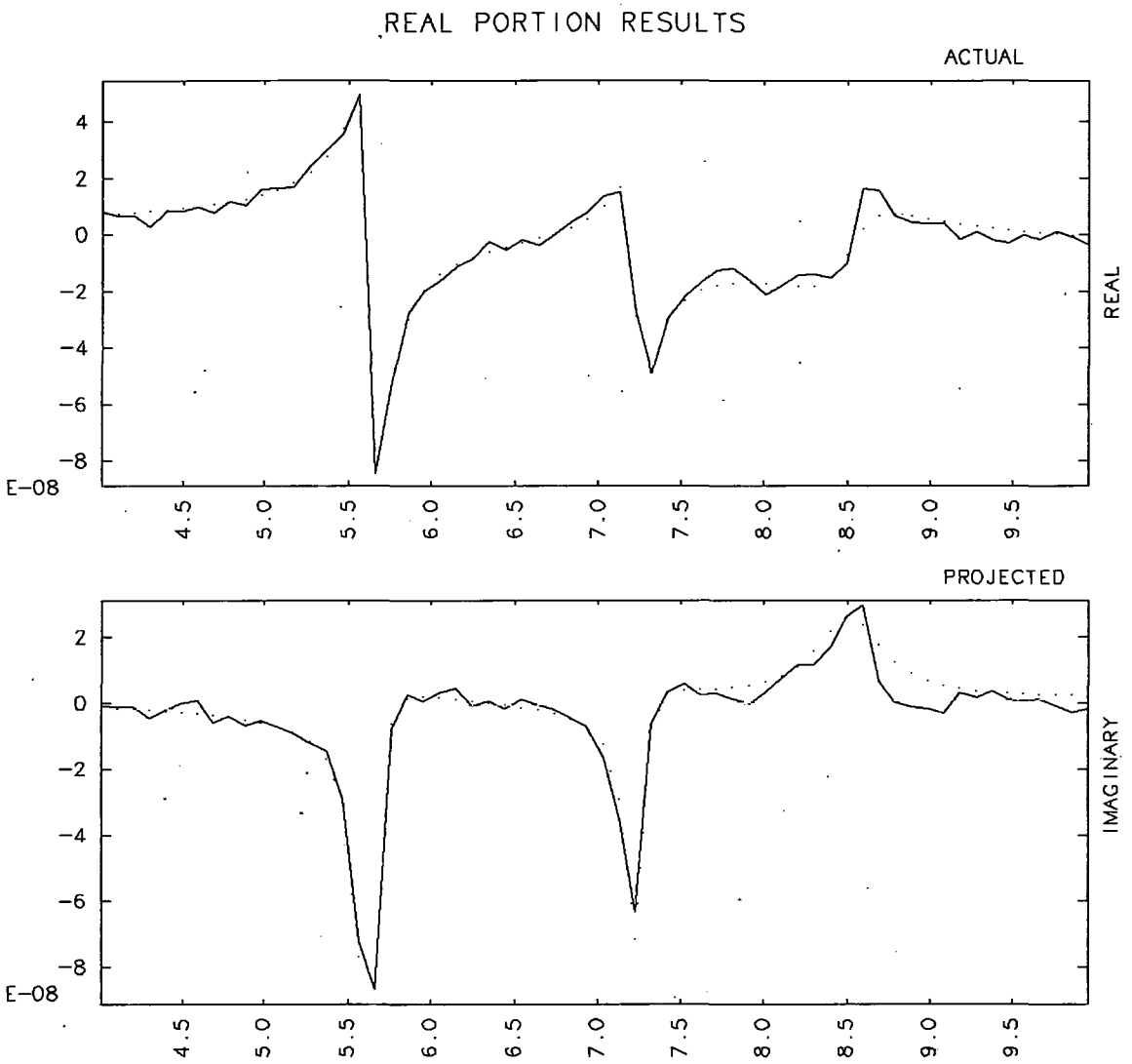


Figure 6. Results Obtained Using Real Portion of Data.

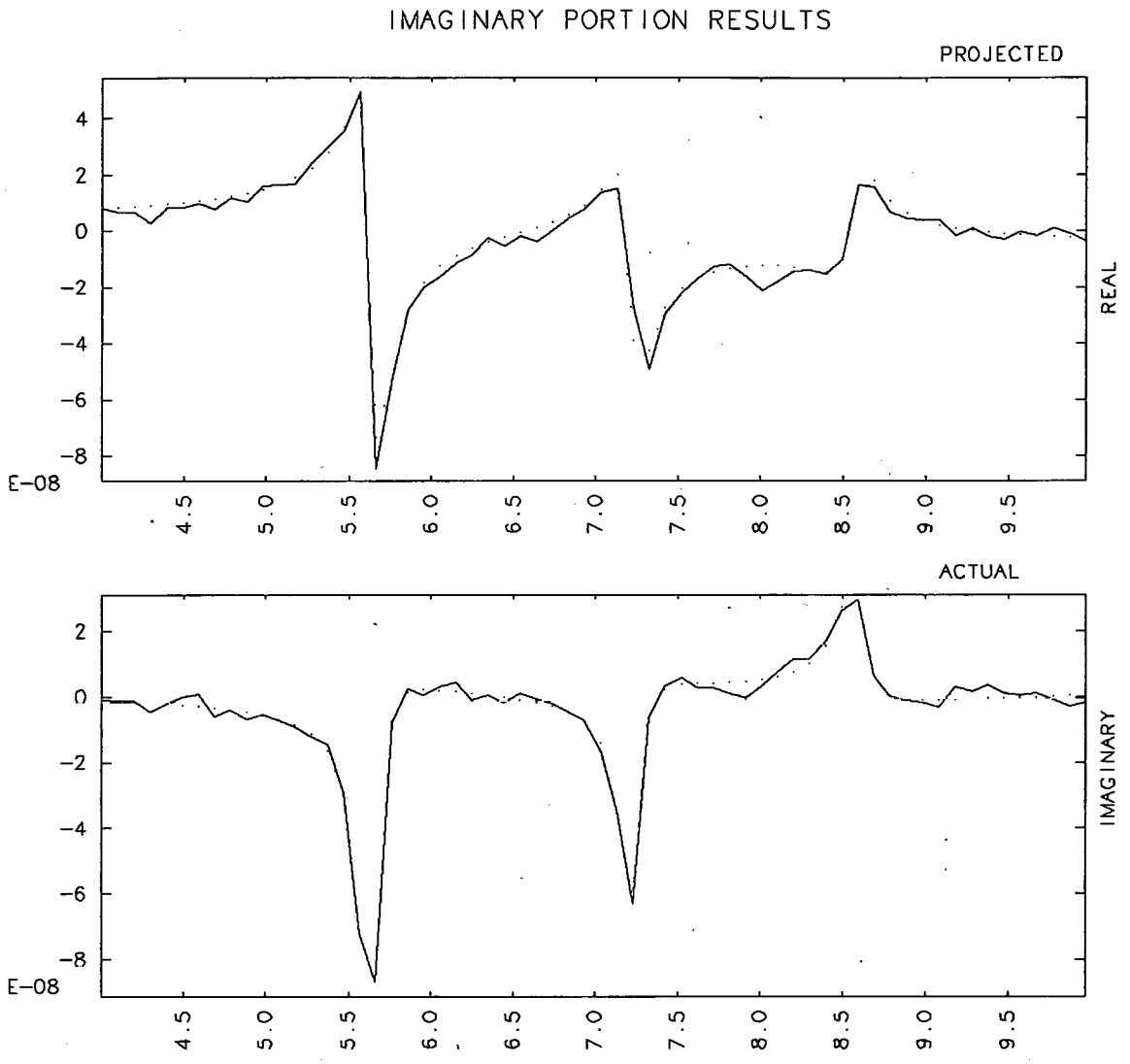


Figure 7. Results Obtained Using Imaginary Portion of Data.

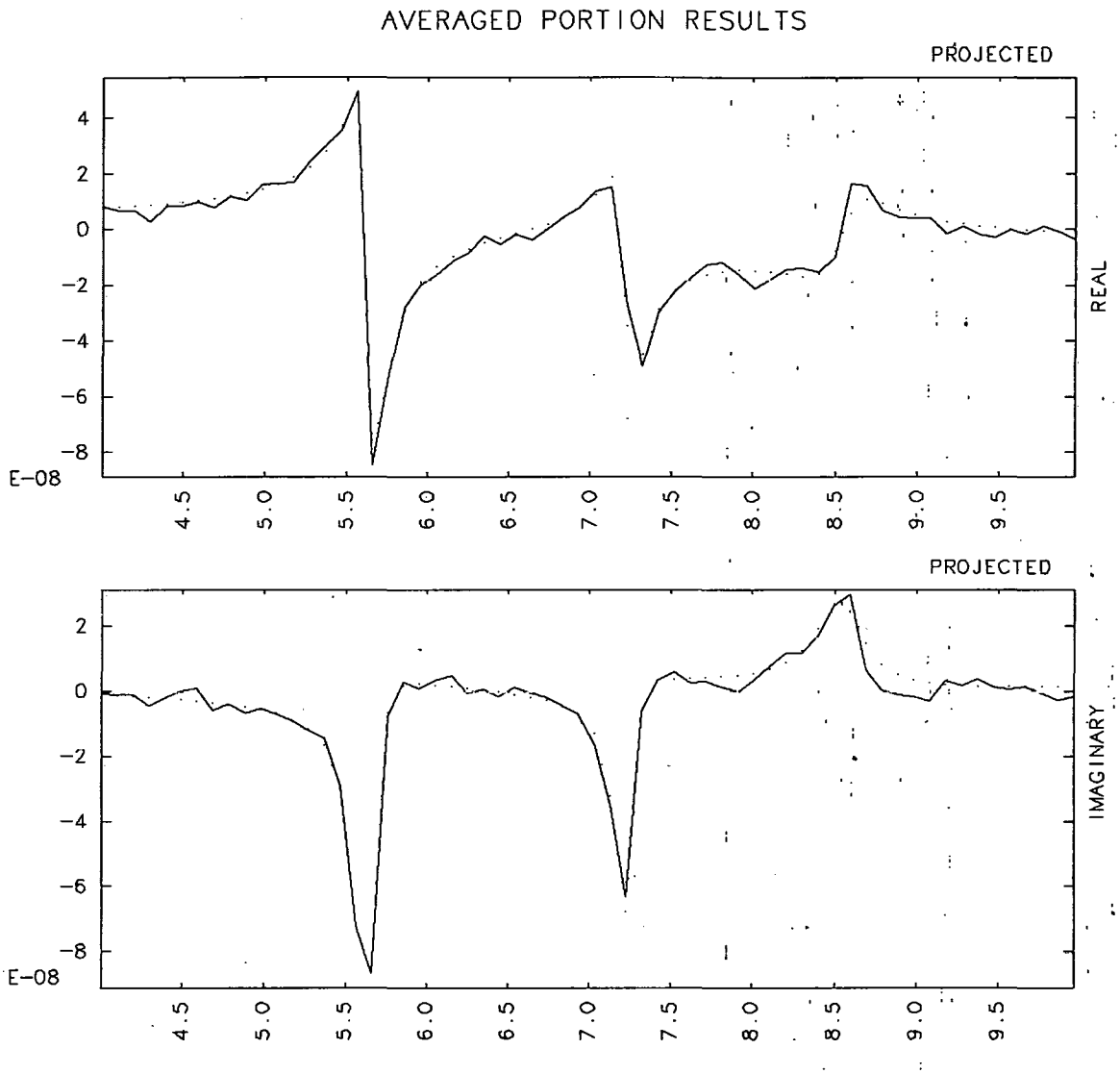


Figure 8. Results Obtained Using Averaged Fit Parameters..

INPUT FILE FORMAT

The data that represents the measured transfer function should be placed in a disk file which will be read in by the TRANSF code. It is assumed by the code that your data function consists of an N-dimensional set of 3-tuples $\{w,H_R(w),H_I(w)\}$ where

w	is the value of the frequency (independent variable),
$H_R(w)$	is the real portion of the function evaluated at w, and
$H_I(w)$	is the imaginary portion of the function evaluated at w.

The first line of the data file must contain three* integers; 1, N, and 0 where N is the number of complex data values. This first line will be read by the code in a format free fashion. Therefore, the three numbers may be placed anywhere between columns 1 and 80 and may be separated by either a comma or a blank space.

The next N lines must contain the data values for the transfer function in 3E20.1 format. Note that this format will allow you to use either E or F format. The only thing that is important is that the values of w must be somewhere between columns 1 and 20, the values of $H_R(w)$ between columns 21 and 40, and the values for $H_I(w)$ between columns 41 and 60.

You may name this disk file whatever you wish since once the code is fired up it will request the file name from you.

At this time the maximum number of data values allowed is 1000. If you exceed this limit then the code will notify you that it will only analyze the first 1000 data values. Also, the maximum number of modes that the code can fit sumultaneously is 10. As it turns out, it is not very efficient to fit more than 6 at a time anyway. If your data contains more modes than this limit then you should use Flag 5 (see section on program control) and subdivide your data into smaller domains.

---

[*] Typically this data is produced by the *FOURIER* code [6] and, consequently, we use this same format for the TRANS code.

PROGRAM CONTROL

In this section we will explain how to run the TRANSF code. In the simplest situation the code is started up and you supply your box number and id along with the name of the disk file that contains your transfer function data. The code then calculates the modal parameters and writes them to both the TTY and a FR80 graphics file along with plots to help you decide on the "goodness" of your fit. In the more general (and common) situation you can instruct the code to deviate from its standard mode of operation and perform additional tasks. This is accomplished by setting control flags which request the code to perform various analysis options. At the present time the following options are available

Flag	Option
1	.... List modal parameters after each iteration,
2	.... Write modal parameters to a disk file,
3	.... Choose frequency starting values manually,
4	.... Suppress bias polynomial terms,
5	.... Fit transfer function over a portion of its original domain,
6	.... Reset number of iterations (default - 10),
7	.... Reset peak search level for starting values (default = 5%)
8	.... Reset all damping start values (default = 3%),
9	.... Reset individual damping start values,
10	.... Constrain value of individual frequency parameters,
11	.... Constrain value of individual damping parameters,
12	.... Convert from old format to new format.

These options will be explained in more detail later in this section. Now, however, we will demonstrate how to use the code in its standard run mode with all of the flags turned off. We first point out that in this and all other demonstrations we will use data that contains 4 sets of resonant or modal parameters. This data was obtained from a SAP IV structural model that represents a nuclear power plant piping system and placed in a disk file called *FEXAMP*. This data file is available on the Cray machine from the take

directory .943794:TRANSF:CRAY:FEXAMP. When you first get started using this code it may be helpful to practice with this file.

To make the standard run we proceed as follows:

```
.....................................................................

transf / .3 .4
Box No and ID
box r67 example run
Reset Flag(s)?
no
Data File
fexamp
modal      8.99147e-08 -3.15815e-08  3.54206e-01  5.63686e+00
modal      5.69396e-08 -3.02993e-08  4.37492e-01  7.23557e+00
modal      1.37301e-08 -8.42792e-09  9.01469e-01  7.90468e+00
modal     -3.24744e-08  1.97721e-08  5.71232e-01  8.58338e+00


     1.685 seconds used for CPU
     0.616 seconds used for I/O
     0.026 seconds used for SYSTEM calls

 all done

.....................................................................
```

The above TTY interactions should be self explanatory. The four sets of numbers that are written to the TTY are the averaged modal parameters $\hat{A}_k$, $\hat{B}_k$, σ_k, and w_k for all four of the modes. A more complete description of the results is written to a FR80 graphics file which contains 6 frames. These are all shown in Figure 9.

The first frame of the output (A) displays both the real and imaginary portions of the input transfer function data. The next frame (B) lists the modal parameters obtained from the real portion curve fit, the imaginary portion curve fit, and the averaged parameters. Note that in this frame AMPR and AMPI correspond to $\hat{A}_k$ and $\hat{B}_k$ respectively (see equations 11 and 12). Also in this frame

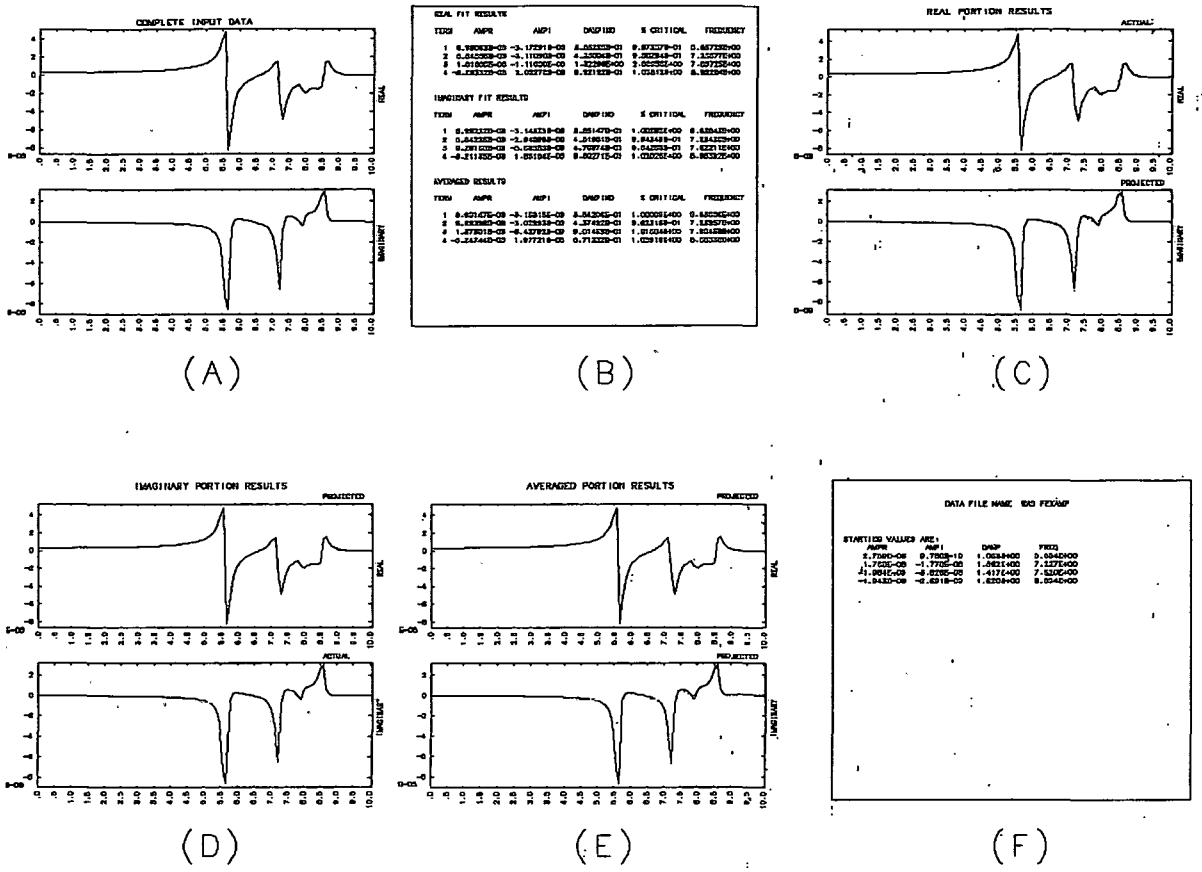


Figure 9. Example FR80 Graphics Output from TRANSF.

the damping is specified in terms of both the σ_k parameter as well as a percent critical value.

The next three frames give a visual presentation of the "goodness" of the fit. Specifically, frame 3 (C) shows the input data as the solid line and the fit produced by the code using only the real portion of the data as the dashed lines. Again we point out here that the imaginary portion of the fit was projected by using the parameters obtained from fitting the real portion of the data only. Frame 4 (D) again shows the input data as the solid line and the fit produced by the code analyzing only the imaginary portion of the data as the dashed line. Frame 5 (E) shows the fit results when the averaged parameters are used.

Finally, frame 6 (F) gives a code summary. Since no flags were set in this case the code summary is rather uninteresting. Normally,

however, when flags are set this will be noted in the summary along with other important control information.

In the remainder of this section we will explain how to use each of the control flags. When the code asks you if you want to reset any of the control flags and your answer is *YES* then the code will ask you *which ones* and prompt you with an asterix (*). You should respond by simply typing the desired flag numbers (see above table) on a single line separated by blanks. If required by that particular analysis option, the code will prompt you for more information.

> *Note that when communicating with the TTY the FOURIER code uses the format free input subroutine FFINC (written by Jim Quigg) which prompts you for data with an asterix (*). Your data can be in either E, F or I FORTRAN format. The last item that you should enter on a line is a period (.) preceded by a blank. If you forget this, the code will probably prompt you for another line. If this is the case then you can simply press the RETURN key.*

FLAG 1 -- LIST PARAMETERS AFTER EACH ITERATION

The curve fitting algorithm in this code is called several times (normally 10) in succession. At each call the modal parameters are updated to improve the fit. If you set Flag 1 then these parameters will be written to the TTY after each update (along with the values of the bias polynomial terms). This option is often quite useful in determining the rate at which the parameters are converging or, with bad luck, diverging.

FLAG 2 -- WRITE PARAMETERS TO DISK FILE

If you set Flag 2 then instead of writing the averaged modal parameters to the TTY at its completion, the code will create a disk file called *TRANOUT* and write them to it. The first line of this file will contain NP (in 1X,I3 format) which is the number of modes used to fit the data. The next NP lines will contain the parameters $\hat{A}_k$, $\hat{B}_k$,

σ_k and w_k in 4(1X,E18.11) format. This option is useful if the modal parameters are going to be subsequently used by some other code for further analysis or processing.

FLAG 3 -- CHOOSE FREQUENCY STARTING VALUES

To determine the frequency parameters from which to start its curve fitting iterations the code automatically searches the imaginary portion of the data to locate the local peaks. It then uses these peak locations as the frequency parameter starting values. If you set Flag 3 the code will let you choose these values yourself. When the code asks; you must tell it how many modes to use and what values you want to use. The code will then use these frequencies to start its iteration process. The better that you guess at these starting values, the better the chances are that the code will converge to the correct answers.

To illustrate the use of this flag, let us direct the code to use 4 nodes and begin the iterations with frequency values of 5.5, 7.2, 7.9, and 8.6Hz. The pertinent TTY interactions are:

```
transf / .2 .3
Box No and ID
box r67 flag 3 example
Reset Flag(s)?
yes
Which ones?
*3 .
Data File
fexamp
Flag 3
No. of Poles
*4
pole  1 Frequency(Hz)
*5.5
pole  2 Frequency(Hz)
*7.2
pole  3 Frequency(Hz)
*7.9
```

```
pole  4 Frequency(Hz)
*8.6

...

all done
```

FLAG 4 -- SUPPRESS BIAS POLYNOMIAL

In the previous section we noted that the code adds second degree polynomials to the transfer function formulas given by equations 11 and 12. The coefficients of this polynomial are included in the curve fitting process. The purpose of this is to help compensate for any offset in the data that could bias the transfer function fit results. If you set Flag 4 then this bias polynomial will not be used.

FLAG 5 -- FIT PORTION OF DATA

Normally the code will work with the entire data file. That is to say, it will perform its curve fitting over the entire frequency domain contained in the input data file. If you set Flag 5 then the code allows you to work with a specified subset of the original domain. When you set this flag the code will respond with

```
Flag 5
Freq Limits   --   Lo  Hi
*
```

You should respond by entering the endpoints of the frequency domain over which you want the code to perform. When you choose this option the code will indicate this subdomain by dashed vertical lines on the first frame of the FR80 output which shows the entire data domain. The subsequent graphs will only show the transfer function over this requested subdomain.

FLAG 6 -- RESET NUMBER OF ITERATIONS

The curve fitting algorithm in this code is called several times in succession and at each call the modal parameters are updated to improve the fit. Normally, the code will iterate the curvre fit 10 times for each the real and imaginary portion. Depending upon the rate of convergence, you may wish to change the number of iterations from 10 to some other value. You do this by setting Flag 6 which will cause the code to prompt you with

```
Flag 6
No. of Iterations?
*
```

You should respond by entering the number of times you wish the code to iterate on the curvefit.

FLAG 7 -- RESET PEAK SEARCH LEVEL

To determine the frequency parameters from which to start the curve fitting iterations the code automatically searches the imaginary portion of the data to locate the local peaks. To do this the code first finds the global absolute value maximum over the data domain. It then sets a cutoff level which is normally chosen to be 5% of this global maximum value. The code will then search out local peaks that are greater than this cutoff. This helps to prevent the code from mistaking small noise fluctuations as resonant frequency peak values. If you wish to change this cutoff level to some other % value of the peak, you would set Flag 7. Upon doing this the code will respond with

```
Flag 7
Peak Search Level in %
*
```

You should respond by entering your desired value.

FLAG 8 -- RESET ALL DAMPING VALUES

To determine the damping parameters from which to start its curve fitting iterations the code uses the starting frequency parameters (see Flag 3 description). That is to say, it chooses these values as per the equation

$$\sigma_k = 2\pi\xi_0 w_k \qquad (k=1,..,N),$$

where, as discussed in a previous section, ξ_0 is a percent of the critical damping values and w_k is the starting frequency parameter. Normally, the code makes the assumption that all the modes have the same value of 3% for ξ_0. If you wish to change this to some other value you must set flag 8. The code will then prompt you with

```
Flag 8
Damping in % Critical
*
```

You should respond by entering your desired value. We again point out here that this one value will be apply to all of the modes. If you wish each mode to use a different value then you would use Flag 9.

For some reason, the code has better luck converging when the starting values of the damping parameters are choosen less than the true values rather than larger. Consequently, if you think the the modes have 5% damping then you should start out with perhaps 3-4%. If on the other hand you choose 6-7%, your chances for convergence are not as good.

FLAG 9 -- RESET INDIVIDUAL DAMPING VALUES

To determine the damping parameters from which to start its curve fitting iterations the code uses the starting frequency parameters (see Flag 3 description). That is to say, it chooses these values as per the equation

$$\sigma_k = 2\pi\xi_0 w_k \qquad (k=1,..,N),$$

where, as discussed in a previous section, ξ_0 is a percent of the

critical damping values and w_k is the starting frequency parameter. Normally, the code makes the assumption that all the modes have the same value of 3% for ξ_0. If you reset Flag 9 then you can choose different values for each mode. When you reset Flag 9 the code will automatically reset Flag 3 and, thus, you must also choose the starting frequency value. To illustrate the use of this flag let us make a code run in which we will use 4 nodes. We will direct the code to choose starting frequency and damping values as follows: 1) 5.5Hz at 2%, 2) 7.2Hz at 3%, 3) 7.9Hz at 2.5%, and 4) 8.6Hz at 3.5%. The pertinent TTY interactions are:

```
.....................................................................
transf / .2 .3
Box No and ID
box r67 flag 9 example
Reset Flag(s)?
yes
Which ones?
*9 .
Data File
fexamp
Flag 3
No. of Poles
*4
pole  1 Frequency(Hz)  Damping(% Crit)
*5.5   2.
pole  2 Frequency(Hz)  Damping(% Crit)
*7.2   3.
pole  3 Frequency(Hz)  Damping(% Crit)
*7.9   2.5
pole  4 Frequency(Hz)  Damping(% Crit)
*8.6   3.5

...

all done

.....................................................................
```

Note that, as we mentioned before (see Flag 8), your chances for

convergence are better if you choose your damping starting values low rather than high.

FLAG 10 -- CONSTRAIN INDIVIDUAL FREQUENCY PARAMETERS

If you set this flag then the code will give you the option of constraining some or all of the frequency parameters. In other words, these paramcters will not be changed at each curve fitting iteration. If you do, in fact, wish to constrain the frequency parameters then you must first choose then manually. Consequently, if you set this flag then the code will automatically set Flag 3 for you. After you have entered the starting values manually (Flag 3) the code will prompt you with

```
Flag 10
Which Freq params to be kept fixed?
*
```

You should respond by entering the mode numbers that you want the code to leave alone. You enter these integer numbers on a single line separated by either a comma or a blank space. Don't forget that your last entry on this lime must be a period (.) preceded by a blank space.

FLAG 11 -- CONSTRAIN INDIVIDUAL DAMPING PARAMETERS

If you set this flag then the code will give you the option of constraining some or all of the damping parameters. In other words, these parameters will not be changed at each curve fitting iteration. If you are going to use this option then the initial, or starting damping parameters must be chosen manually. You can accomplish this by either using Flag 8 or Flag 9. After you have entered the starting values manually (Flag 8 or 9) the code will prompt you with

```
Flag 11
Which Damp params to be kept fixed?
*
```

You should respond by entering the mode numbers that you want the code

to leave alone. You enter these integer numbers on a single line separated by either a comma or a blank space. Don't forget that your last entry on this lime must be a period (.) preceded by a blank space.

FLAG 12 -- CONVERT DATA FILE TO NEW FORMAT

This option is made available especially for people who were users of the old 7600 version of the TRANSF code. If you are one of these users and have data files that are in the old format then this option will automatically convert your old file to a new one which is in the proper format. Remember, if you are using the TRANSF code on a CRAY machine and want to run it on an old 7600 file then you must first convert it to a CRAY 8 bit format file. To do so you use the TRANS Utility Routine (See LTSS-210 or the Summary Sheets LCSD-50). The basic steps are:

1) Use XPORT to ship file to CRAY from 7600 (or storage)

2) Use the Utility Routine TRANS to convert from a 6 to 8 bit format as follows:

TRANS I=(*Oldfile*,7600) O=(*Newfile*,Cray)

You are now ready to run TRANSF on the data. The code will first prompt you with

FLAG 4
OLDFILE

You should respond by entering the name of your old file (which has already been converted to 8-bit format if you are on a CRAY machine). Next the code will prompt you with

NEWFILE NAME

You should respond by entering the name that you want the newfile to be called.

To illustrate the use of this option let us assume that we have a file called FEXOLD in the old format and that we want to run the new version of TRANSF on it. The following TTY interactions illustrate how

```
.....................................................................
transf / .2 .3
box no and id
box r67 flag 12 example
reset flag(s)?
yes
which ones?
*12 .

flag 12
oldfile
fexold
newfile name
fexnew
 1 function(s) from fexold  written to fexnew

data file
fexnew

...

all done

.....................................................................
```

REFERENCES

[1] Weaver, H.J., *Applications of Systems Theory to Structural Analysis*, Report No. UCRL-53094, Lawrence Livermore National Laboratory, July 1981.

[2] Cooley ,J.W. and J.W. Tukey, An Algorithm for the Machine Calculation of Complex Fourier Series, Math. Computations, Vol. 19, April 1965.

[3] Weaver, H.J., *Discrete and Continuous Fourier Analysis*, To be published. Currently available as Report No. UCIR-998, Lawrence Livermore National Laboratory, January 1980.

[4] Raven, F.H., *Automatic Control Engineering*, McGraw HIll Book Co., New York, 1968.

[5] Richardson, M.H., *Detection of Damage in Structures from Changes in their Dynamic (Modal) Parameters – A Survey*, Report prepared for LLNL by Strucutral Measurement Systems Inc., Available from Lawrence Livermore National Laboratory as Report No. UCRL-15103, 1979.

[6] Weaver, H.J., *FOURIER Code User Manual*, Report No. UCID-19237, Lawrence Livermore National Laboratory, Septemper 1981.

[7] Bevington, P.R., *Data Reduction and Error Analysis for the Physical Sciences*, McGraw-Hill Book Co., New York, 1969.

[8] Weaver, H.J. et al., *Analytical Qualification of System Identification (Modal Analysis) Codes for use in the Dynamic Testing of Nuclear Power Plant Structures*, Report No. UCID-18144, Lawrence Livermore National Laboratory, January 1980.

DISCLAIMER

Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore Laboratory under Contract W-7405-Eng-48.

box r67 joe weaver

tv80lib nips processr 07/14/78

STARTED 07:45:54r 12/08/81
Finished 07:50:06. 41 frames plotte