INTERNAL REPORT SDC-94-**

BACKGROUND AND DATA REDUCTION ON THE SDC VISAR

G. F. Raiser

October 1994

Shock Dynamics Center

Washington State University

Pullman, WA 99164

I. Introduction

The purpose of this report is to give the reader a working knowledge of the general theory behind VISAR [1] data reduction and to provide a reference for the program 'SDCVIS.FOR' created to implement this theory in steps. In order to do this, some brief background material on the theory behind a velocity interferometer is presented first, then the effects of using a 'window' material attached to the monitored surface is covered, and finally the extension of the theory to VISAR systems is discussed. The final form of the equation relating the oscilloscope records from the VISAR to the desired particle-velocity history is summarized along with its assumptions and limitations. The data reduction procedure is then covered in detailed step-by-step fashion for reducing sample data from an actual experiment. The correction factors for window materials as well as the actual program list are given in the appendices.

II. Theoretical Background for VISAR Data Reduction

A. Velocity Interferometry Theory

In this first section, the relation between the number of fringes observed from a velocity interferometer and the particle velocity of the monitored surface will be derived. Since a VISAR is a velocity interferometer [2] with certain modifications, a derivation of the velocity-interferometer equation relating fringe count to particle velocity is done first. Section B. will cover the modification of the theory to account for a window material attached to the monitored surface. Section C. will cover the extension to the VISAR and the overall results will be summarized in Section D.

A schematic of the velocity interferometer is shown in Figure 1. Half of the light which enters the photomultiplier tube (PMT) travels the path *abcdefghi*, while the other half travels the path *abcdehi*. Therefore, the PMT observes a superposition of one beam that was at the reflecting surface at time t along with another beam that was there at time t-τ, where τ is the difference between the two path lengths divided by the speed of light. The beam intensity at the PMT (and therefore its voltage output on an oscilloscope) will remain constant if the frequency of the two beams is the same and will vary if the frequencies are different. The frequency of a given beam will be Doppler shifted from its original frequency if the surface is moving. If the surface is moving with constant velocity for a given time interval, then at any two times $(t-\tau,t)$ within this interval, the two beam frequencies of the interferometer are the same, the combined intensity remains constant, and the PMT's oscilloscope record registers a constant voltage. If, however, the velocity is decreasing or increasing, each beam will be Doppler shifted by a different amount (according to the velocity of the surface at the times t and t-τ) and the intensity at the PMT will vary, resulting in 'fringes' in the oscilloscope record of its output. As shown below, each fringe will then correspond to a given *change* in velocity, and by counting the fringe increments of the oscilloscope record, one can directly calculate the velocity of the surface at any time.

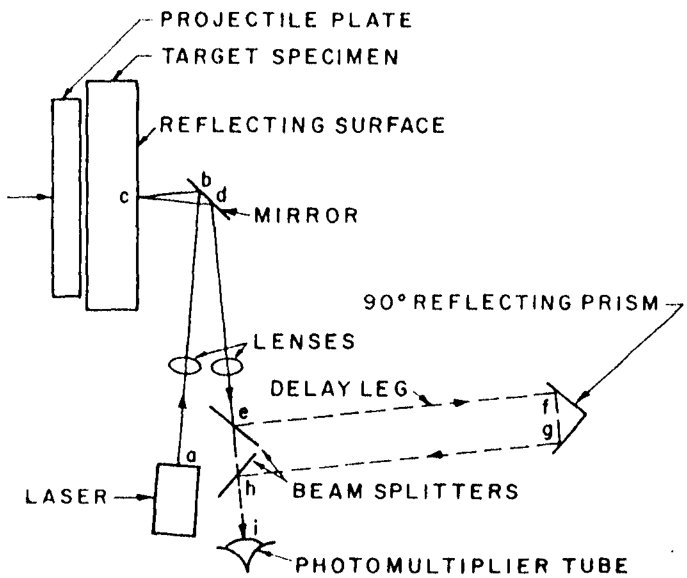


Figure 1 - The Velocity Interferometer (from [3])

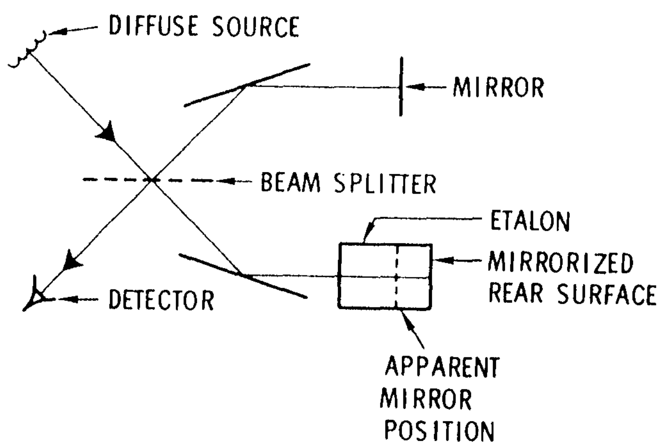


Figure 2 - The Wide-Angle Michelson Interferometer (from [1])

The discussion which follows is taken from Barker and Hollenbach [2] and Clifton [3]. Their analyses are combined here to provide a simple, yet complete derivation of the fringe count-particle velocity relation. The velocity interferometer produces a fringe frequency $\nu_{V.I.}(t)$ resulting from a combination of one beam Doppler shifted at the current light frequency $\nu_L(t)$ and another beam Doppler shifted at a time τ earlier, call it $\nu_L(t-\tau)$. In equation form:

$$\nu_{V.I.}(t) = \nu_L(t) - \nu_L(t-\tau). \tag{2.1}$$

In a Michelson interferometer, the fringe frequency $\nu_m(t)$ results from combining light at the original laser light frequency ν_{L0} with light that was Doppler shifted by the moving surface $\nu_L(t)$:

$$\nu_m(t) = \nu_L(t) - \nu_{L0}. \tag{2.2}$$

If the beam is normal to the moving surface, then $\nu_m(t)$ is given by [4]:

$$\nu_m(t) = \frac{2u(t)}{\lambda_o\left[1 - \frac{u(t)}{c}\right]}, \tag{2.3}$$

where $u(t)$ is the surface particle velocity, λ_o is the original wavelength of the laser light, and c is the speed of light. In plate impact experiments, u/c is on the order of 10^{-5}. Simplifying then gives

$$\nu_m(t) = \frac{2u(t)}{\lambda_o}. \tag{2.4}$$

Combining eqns. (2.1-2.2) gives:

$$\nu_{V.I.}(t) = \nu_m(t) - \nu_m(t-\tau). \tag{2.5}$$

It is obvious from this equation that the velocity interferometer can be considered as two Michelson interferometers shifted in time by the delay time τ. The fringe count $F(t)$ from an oscilloscope record would simply be the integral of the velocity interferometer frequency up to time t:

$$F(t) = \int_0^t \left[\nu_m(\tilde{t}) - \nu_m(\tilde{t}-\tau)\right] d\tilde{t}, \tag{2.6}$$

with the assumption that $\nu_m(\tilde{t}) \equiv 0$ for $\tilde{t} < 0$. Using this fact allows an equivalent expression to be written as:

$$F(t) = \int_{t-\tau}^{t} \nu_m(\tilde{t}) d\tilde{t}. \tag{2.7}$$

Taking the relation for $\nu_m(t)$ from equation (2.4) and substituting here gives:

$$F(t) = \frac{2}{\lambda} \int_{t-\tau}^{t} u(\tilde{t}) d\tilde{t}. \tag{2.8}$$

Denoting $U(t)$ as a displacement of the monitored surface at time t, this equation is equivalent to writing:

$$F(t) = \frac{2}{\lambda}\left[U(t-t_1) - U(t-t_1-\tau)\right], \tag{2.9}$$

where t_1 is the time required for light to travel from the monitored surface to the PMT via the direct path *abcdehi* (Figure 1). As seen from this equation, the velocity interferometer is more accurately described as a "differential displacement interferometer," because the fringe count at time t gives the displacement which occurs in the time interval $(t,t\text{-}\tau)$. Moreover, if the time t_1 is taken to be zero (thereby defining $t = 0^+$ as the first time when fringes are observed), then for times t less than τ, $U(t-\tau)$ is zero and this equation reduces to

$$F(t) = \frac{2}{\lambda} U(t). \tag{2.10}$$

Thus, for the time interval $0 < t < \tau$, the velocity interferometer behaves as a Michelson interferometer.

Consider again equation (2.8). It is exact, except for the restriction that $u/c \ll 1$. Ideally, once a value of the fringe count is given at a time t from the oscilloscope record, this equation could then be used to ascertain the particle velocity at some time t'. By the mean value theorem, this equation can be written as

$$u(t-\theta\tau) = \frac{\lambda F(t)}{2\tau}, \tag{2.11}$$

where θ lies in the interval (0,1). Exact expressions for $u(t)$ have been derived [3] for three separate cases where, within the interval $(t-\tau, t)$, the particle velocity is constant, the acceleration is constant, and the jerk is constant. The results show that the equation

$$u(t-\tau/2) = \frac{\lambda F(t)}{2\tau}, \tag{2.12}$$

which is exact for the case of constant acceleration in this interval, "has the advantage of providing convenient data reduction, [and] is sufficiently accurate for times greater than

$\tau / 2$." This is then the chosen relation between *F(t)* and $u(t-\tau/2)$ used for the velocity interferometer.

From eqn. (2.12) it is seen that, for a given particle velocity *u*, the number of fringes observed will be large if τ is large. Since τ can be easily adjusted through increasing or decreasing the delay leg, the velocity interferometer is extremely flexible as far as setting its sensitivity to particle velocities. One drawback to a large value of τ occurs during periods where the velocity is changing rapidly (e.g. during an initial jump in particle velocity). Here, the instrumentation often will not be able to resolve such high-frequency changes in light intensity from the interferometer.

One drawback to the original velocity interferometer is its inability, in some cases, to properly identify 'reversals', which are changes from velocity increasing to velocity decreasing or vice versa. The amibiguity arises when the recorded light level is at a maximum or minimum just prior to the reversal. This can be corrected with the addition of some additional optical components, which is done for the VISAR. These issues are discussed in section C.

B. Modification of Theory for Window Considerations

One modification to the formula given above becomes necessary for the case where the monitored surface is not a free surface, but is on the other side of a window material, through which the laser light passes. The equation derived earlier for the fringe count is:

$$F(t) = \int_0^t \left[\nu_m(\tilde{t}) - \nu_m(\tilde{t} - \tau)\right] d\tilde{t}, \tag{2.13}$$

which is equivalently expressed as:

$$F(t) = \int_{t-\tau}^{t} \nu_m(\tilde{t}) d\tilde{t}. \tag{2.14}$$

Here, ν_m is the frequency of light entering the interferometer. If the monitored surface is a free surface moving at a velocity *u*, and $u/c \ll 1$ then the value of ν_m (call it ν_o) is given by (eqn 2.4):

$$\nu_o = 2u / \lambda. \tag{2.15}$$

If, however, the light travels through a window material to get to the surface moving at velocity *u*, then it experiences an additional frequency shift (call it $\Delta\nu$). The reason is that as the window material is strained under stress-wave propagation, its index of refraction changes and this results in a frequency shift. One can show that [5], if the window material obeys the Gladstone-Dale model

$$d\rho / \rho = dn / (n-1), \tag{2.16}$$

then the frequency ν_m is equal to ν_o (there is no shift in the frequency). Thus, the fractional difference, $\Delta\nu / \nu_o = (\nu_m - \nu_o)/\nu_o$, is a measure of the material's degree of departure from this model.

What is needed at this point is an equation which handles this frequency change, yet still relates the fringe count to the actual particle velocity of the monitored surface. By the definition of ν_o above, the particle velocity is given by $u(t) = \frac{1}{2}\lambda\nu_o(t)$. Barker and Hollenbach [2] argue that $\Delta\nu / \nu_o$ is a single-valued function the particle velocity which can be experimentally determined, and since $\nu_o \equiv \nu_m / (1 + \Delta\nu / \nu_o)$, the equation for *u(t)* becomes

$$u(t) = \frac{\frac{1}{2}\lambda\nu_m(t)}{1 + \Delta\nu / \nu_o}. \tag{2.17}$$

Solving for $\nu_m(t)$ and substituting this into the equation for the fringe count (eqn. 2.14) gives

$$F(t) = \frac{2}{\lambda}\int_{t-\tau}^{t} u(\tilde{t})(1 + \Delta\nu / \nu_o)d\tilde{t}. \tag{2.18}$$

Evaluation of $(1{+}\Delta\nu / \nu_o)$ for various window materials shows that this factor is a slowly varying function of *u*. Also, *u(t)* seldom changes appreciably in the time interval $(t, t-\tau)$. This factor can then be removed from the integrand and evaluated at the average value of *u* in the time interval $(t, t-\tau)$, which, as discussed before, is sufficiently approximated as $u(t - \tau/2)$. The resultant equation then becomes

$$u(t - \tau / 2) = \frac{\lambda F(t)}{2\tau(1 + \Delta\nu / \nu_o)}, \tag{2.19}$$

which is recognized as the final form of the data reduction equation used for the velocity interferometer. Of course, since the factor $(1{+}\Delta\nu / \nu_o)$ depends on *u*, an iteration procedure is necessary in the program to zero in on the correct *u*. Barker and Hollenbach's [2] experimental data on PMMA, fused silica and sapphire, along with Wise's data [6] on lithium fluoride are summarized in Appendix A. These data were numerically fitted and then implemented into the data reduction program.

C. Extension to VISAR

There are two major drawbacks to the velocity interferometer . First, it requires a mirror finish on the monitored surface. If the polish is not good, then the two beams which are split from this diffuse source will continue to diverge as they travel the different path lengths of the interferometer. When they are finally recombined, they will have lost spatial coherence. The result is poor contrast in the observed signal. The second drawback is its sensitivity to tilting of the monitored surface during the motion of interest. Because of the path difference in the two legs, the two beams are shifted by a different amount for a given tilt. When they are recombined at the beamsplitter, they no longer completely overlap, which results in a drop in signal strength at the photodetector. Barker and Hollenbach recognized that to solve these problems, a modification was needed "such that the two legs of the interferometer appear equal, yet such that the light in one of the legs is delayed with respect to the light in the other leg." This can be accomplished with a wide-angle Michelson Interferometer (WAMI) shown schematically in Figure 2.

Zwick and Shepherd [7] analyzed this type of interferometer and showed that, in order to maintain a constant path difference between light in the two legs for an arbitrary (but small) incident angle on the mirrors, the mirror in the air leg must be closer to the beamsplitter (or, the mirror in the etalon leg must be farther away from the beamsplitter) by an amount

$$x = h(1 - 1/n), \tag{2.20}$$

where h is the length of the etalon and n is its index of refraction. This restriction is interpreted as the condition for collinearity of the recombined beams, called the "wide-angle criterion." What it essentially means is that light entering the beamsplitter must behave as if it travels the same distance in either leg, thereby making collinearity of the recombined beams independent of the initial beam's angle of incidence on the beamsplitter. This ensures insensitivity to both tilt and divergence of the incoming beam, allowing light from diffuse sources and from optical fibers to be used in the interferometer. Since the light in the etalon leg travels a longer total distance and travels slower inside the etalon, it is delayed with respect to light in the other leg by an amount:

$$\tau = \frac{2h}{c}(n - 1/n). \tag{2.21}$$

Different lengths of etalons can be chosen depending on the maximum anticipated particle velocity to be measured and the number of fringes desired. Twelve inches of BK-7 etalon will produce a delay time of 1.75 nsec, and one fringe will correspond to about 0.147 mm/μsec (λ=514.5 nm and neglecting window considerations). This places some restrictions on the ability to measure very low particle velocities. In such cases, a lens delay system is often used in place of the etalon [8].

One additional drawback to the use of the velocity interferometer of Figure 1 becomes evident during data reduction. The fringe count $F(t)$ used in eqn. (2.19) is obtained from the oscilloscope record; however, when the PMT's output voltage is at a maximum or minimum, it is impossible to tell whether or not there is a 'reversal' (i.e. a change from velocity increasing to decreasing or vice versa), because the oscilloscope record looks the same in either case. If, however, there is another record available which contains the same information but is 90° phase-separated, then its voltage is at the mean in such cases and a reversal (voltage increasing to decreasing or vice versa) is clearly distinguishable. The VISAR system developed by Barker and Hollenbach [1] utilizes the beneficial properties of the WAMI, but also contains additional optics that allow recording of two such signals. Signals in 'quadrature' can also be arranged for velocity interferometers with the addition of some optical components.

The VISAR developed in the Shock Dynamics Center is shown in Figure 3. Laser light is sent through lenses L1 and L2 to shrink its diameter before going through a small hole in the center of mirror M2. The light is then focused through lens L3 into an optical fiber which runs to the target assembly. The beam exiting the fiber is focused down through the window material onto its diffusely-reflecting face. Reflected light from this surface is collected back into the fiber and returns through lens L3 as a larger diameter, nearly collimated beam to mirror M2. This beam will first travel through a sapphire plate, where its back-reflected light is collected into a photomultiplier tube (PMT) for purposes of monitoring the beam intensity (BIM). Half of the beam travels through the etalon-delay leg, while the other half travels twice through the $\lambda/8$ wave plate, causing a phase shift between the S and P components of the light. The beams are recombined and sent through the polarizing beam splitter, where the reflected P-component is collected into one PMT and the transmitted S-component is collected into the other. Both will record the same information, but will be phase-separated (typically set near 90° by rotating the $\lambda/8$ wave plate) for the reasons described above.

One final modification to equation (2.19) is necessary for it to be applicable to a VISAR. Barker and Schuler [9] recognized that the small (less than one part in 10^5) change in wavelength of the Doppler-shifted light entering the VISAR results in a very slight change in the index of refraction of the etalon. This change will then affect the fringe count observed at the photodetector. Their analysis shows that this index change can be approximated and the contribution to the fringe count factored into the data reduction equation:

$$u(t-\tau/2)=\frac{\lambda F(t)}{2\tau(1+\delta)(1+\Delta\nu/\nu_o)}. \qquad (2.22)$$

Here, δ is a function only of the etalon's index of refraction characteristics and the wavelength of laser light used in the experiments. For the SDC VISAR, a value of δ =0.036 for BK-7 glass at λ=514.5 nm is used [10].

C. Summary of VISAR Theory for Data Reduction

The final form of the VISAR data reduction formula is

$$u(t-\tau/2)=\frac{\lambda F(t)}{2\tau(1+\delta)(1+\Delta\nu/\nu_o)}, \qquad (2.23)$$

where:

λ is the wavelength of the laser light

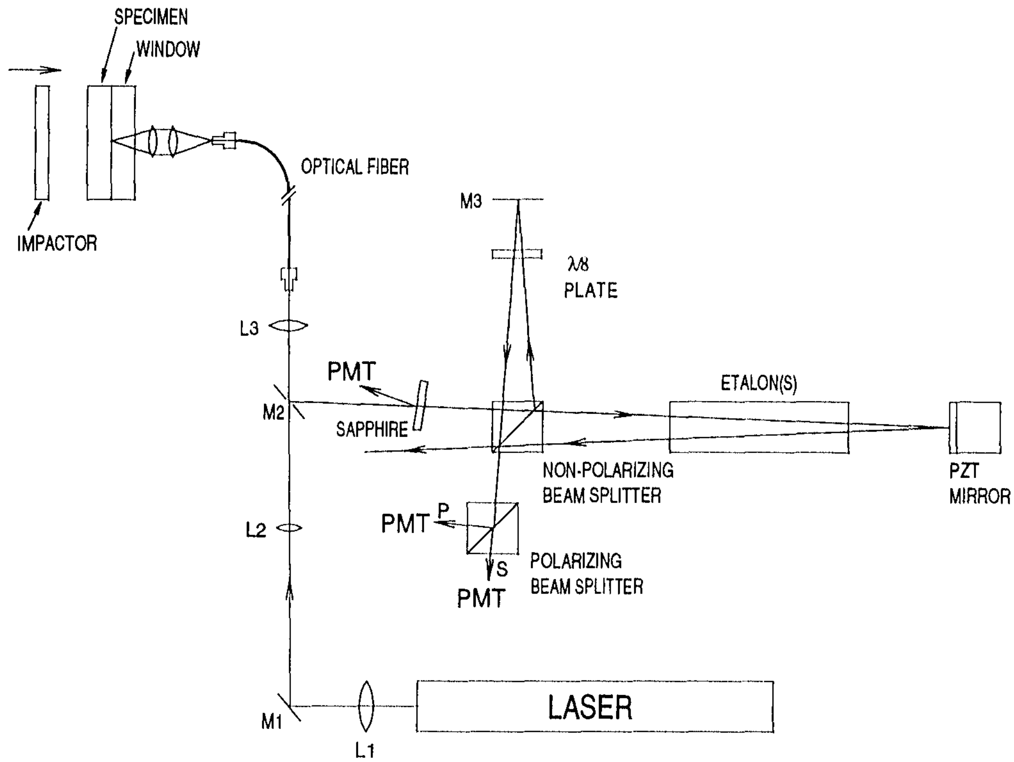


Figure 3 - Representative schematic of the Shock Dynamics Center VISAR

F(t) is the fringe count of the oscilloscope record

τ is the delay time, $\tau = (2h/c)(n - 1/n)$, where h is the etalon length, n is its index of refraction, and c is the speed of light

δ is the correction factor for dependence of the etalon's index of refraction on the wavelength of the Doppler-shifted light (for the SDC VISAR, δ=0.036 when λ=514.5 nm)

$\Delta\nu / \nu_o$ is a correction for the use of window materials which accounts for the window's change of index of refraction as it is strained (see Appendix A for these values as functions of the particle velocity).

Often, $\Delta\nu / \nu_o$ is assumed constant and all the factors but *F(t)* are lumped into a "fringe constant":

$$K_e = \frac{\lambda}{2\tau(1+\delta)(1+\Delta\nu/\nu_o)}. \tag{2.22}$$

The following assumptions/limitations apply to the VISAR data reduction formula:

The earliest part of the fringe record (for a time interval of τ) is when the VISAR acts as a Michelson interferometer. The equation above is valid only after this interval. If a large velocity jump is presumed in this interval, then one can manually add a velocity equal to an integer number of fringes times the fringe constant to the entire record starting at time $\tau/2$.

The particle velocity calculated on the right side of this equation is, by the mean value theorem, strictly equal to $u(t-\theta\tau)$, where θ lies in the interval (0,1). In practice, $\theta = 1/2$ is chosen as a sufficiently accurate value. Clifton [3] shows this to be a good approximation.

The contribution of wavelength change of the laser light on the fringe count is assumed to be negligible, except for its contribution to the refractive index change of the etalon. This is justified because, for shock wave experiments, $\Delta\lambda / \lambda_o$ is of the order of 10^{-5}.

The ratio $\Delta\nu / \nu_o$ is assumed to be a single-valued function of u until arrival of the first wave at the free surface of the window material. This is verified for PMMA and fused silica [2].

The factor (1+δ) is assumed to completely account for the etalon's change in index of refraction during motion of the monitored surface (Barker and Schuler [9] derive this and verify it experimentally).

III. Step-By-Step Data Reduction - An Example

A. Introduction

This particular section will cover the specifics of reducing data from a VISAR experiment. Some of the ideas are taken from Barker's data reduction scheme [11]. Background on what files are needed, the use of graphics for viewing data and how to determine the phase difference between the two signals are all covered before going on to the main data reduction program (listed in Appendix B). Each step is described along with the proper keyboard sequence for executing that step on sample data provided with the program. This data is from an actual experiment and helps to guide the reader through the data reduction process.

Before starting, the following files should be included in one directory (currently, these files are included in the directory 'c:\visar' on the Gateway 2000):

sdcvis.exe	-the main data reduction program
v1.dat	-the first sample VISAR trace
v2.dat	-the second sample VISAR trace
b.dat	-the sample beam intensity monitor trace
pre1.dat	-the first pre-experiment sample trace
pre2.dat	-the second pre-experiment sample trace
polar.spw	-the SigmaPlot file used for polar plotting pre1.dat and pre2.dat
polar.xfm	-the Math Transform file used to fit an ellipse to the polar plot

A graphics routine has been included into the program, although a commercially available software package for this purpose would be more flexible. In order to facilitate the use of any additional software of this nature, the program is compiled to run in a windows environment, and routines which output the data to files are included. This allows the user to easily switch back and forth between the main program and the desired software. At any time during the data reduction process, the user can call up the graphics software in a separate window and view the data. One graphics program chosen to illustrate this process is 'SigmaPlot.' All necessary steps in using SigmaPlot will be given in the discussion below.

B. Determination of the Phase Shift

The phase difference between two VISAR signals should be set as near to 90° as possible before the experiment. A slight offset from this angle is acceptable as long as an estimate of the actual phase difference can be made. The following procedure describes one approach for calculating this difference.

With the PZT operational, a pre-experiment recording of the VISAR traces is made. A graph of these two recordings is shown in Figure 4. If the voltage of one trace is plotted

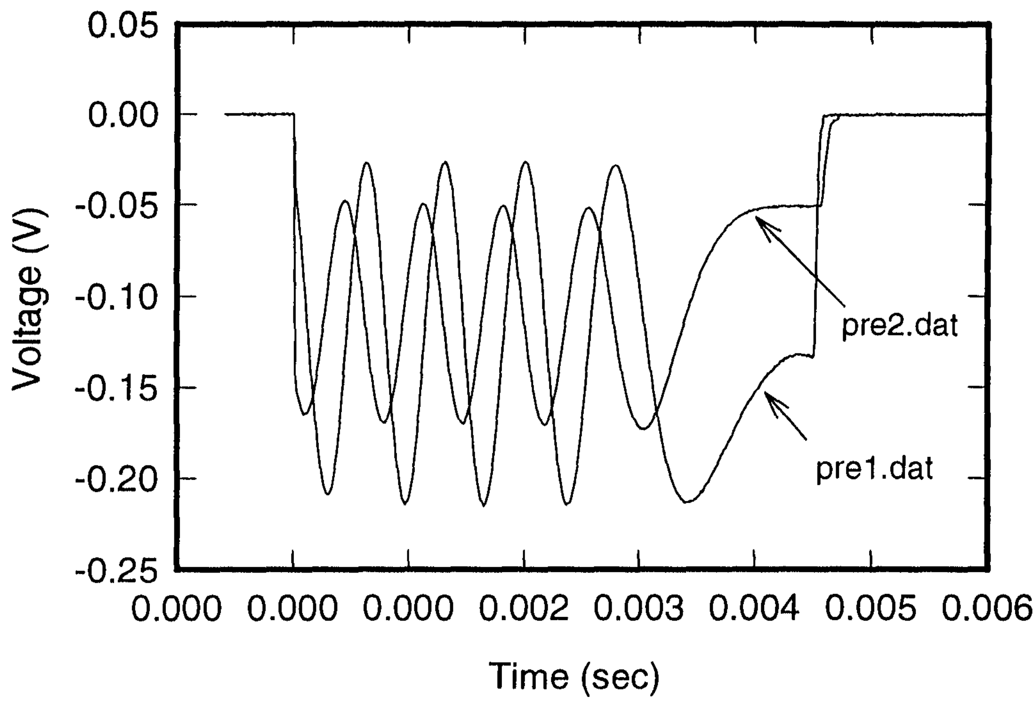


Figure 4 - Pre-experiment VISAR traces.

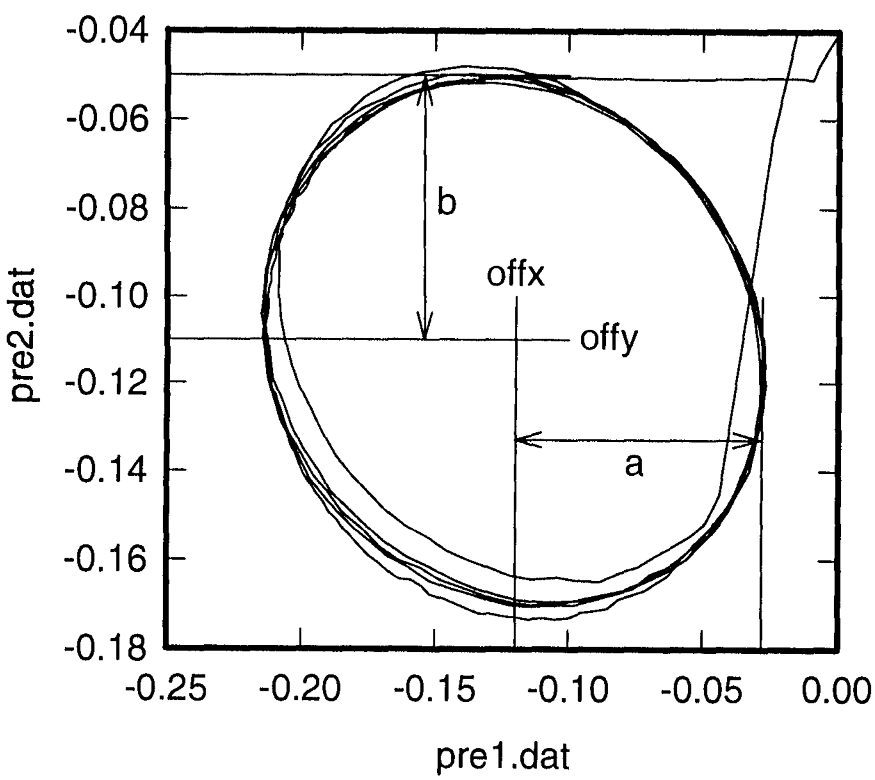


Figure 5 - Polar plot of the pre-experiment VISAR traces.

against the voltage of the other, the shape resembles an ellipse, as seen in Figure 5. At any given time, the voltages of these traces can be represented as:

$$V1 = \mathrm{offx} + a\sin\theta$$

and

$$V2 = \mathrm{offy} + b\sin(\theta + \varphi).$$

This is assuming, of course, a constant offset and constant amplitude, which is nearly the case for these two recordings. The procedure for finding φ is simply to first construct and overlay onto the data a separate plot of an ellipse using these formulas and estimates for offx, offy, a, b, and φ. Then, through successive observations of the two plots, adjust these parameters until the constructed ellipse most nearly fits the experimental one.

With the PC running in windows mode, do the following:

Find the SigmaPlot icon and double-click on it.

Click-and-hold on 'File' pull-down menu. Select 'Open...' and change the active directory to 'c:\visar.'

Double-click on the file 'polar.spw.'

Click-and-hold on 'File' pull-down menu. Select 'Import Data' and change the active directory to 'c:\visar.'

Double-click on the file 'pre1.dat'

Click 'OK' to the Import Warning.

Choose 'Import' to write the data into columns 1 and 2.

SigmaPlot will import the first pre-experiment trace into columns 1 and 2.

Move the cursor by clicking once on the box in the first row, third column of the worksheet.

Click-and-hold on the 'File' pull-down menu. Select 'Import Data.'

Double-click on the file 'pre2.dat'

Click 'OK' to the Import Warning.

Choose 'Import' to write the data into columns 3 and 4.

To observe the polar plot of the two traces:

Click the page button on the toolbar (it looks like a paper with a graph on it that has one corner folded down).

In order to curve-fit this data with an ellipse as described above, a math transform file was written. Note that the angles used by the transform have already been entered into column 5 of the worksheet. To fit the data,

Click-and-hold on the 'Math' pull-down menu. Select 'Transforms...'

Select 'Open...' and make sure the current directory is 'c:\visar.'

Double-click on the file 'polar.xfm.'

Click on the 'Execute' button. To view the results, click the page button on the toolbar.

The result at this point should look like Figure 6. The magnifying glass buttons on the toolbar can be used to zoom in or out on the graph. By zooming in close enough, the dotted curve representing the fit to this data becomes evident. From the transform file, it is seen that the phase angle which fits this data is approximately -100°. When executing the program, the phase difference will be either -100° or +100°, depending on whether the phase of the first trace is initially ahead or behind the second. If the phase of the first trace is initially ahead of the second, then choose the negative value. Regardless, if the wrong value is chosen, it will become evident in the velocity profile.

The reader is encouraged to practice altering the parameters in order to see the effects. When done:

Click-and-hold on the 'File' pull-down menu and select 'Exit.'

Click on the 'No' button in order to keep the original files the same.

C. Execute Main Program

The PC should be running windows prior to executing the program. To start the program, do the following:

Find the 'File Manager' icon and double-click on it.

Make sure it is displaying the necessary files in the proper directory (currently, 'c:\visar').

Double-click on the file 'sdcvis.exe' to execute the program.

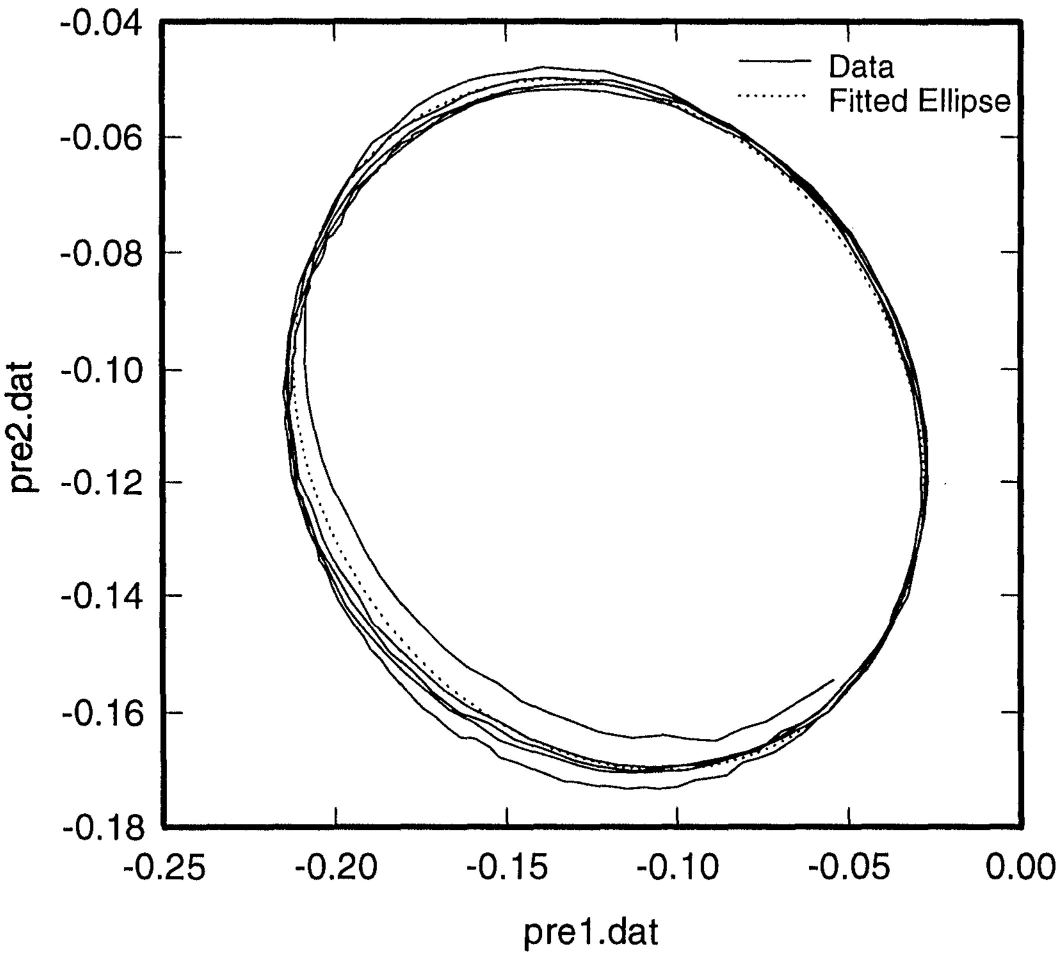


Figure 6 - Polar plot of pre-experiment VISAR traces and fitted ellipse.

Maximize the window by clicking the upward-pointing arrow in the upper right corner of the program's window.

Another way to execute the program is to double click on the 'VISAR Data Reduction' icon from the windows environment.

In the following, the required user input is in **bold** type.

What is the filename of VISAR signal #1?
v1.dat
What is the filename of VISAR signal #2?
v2.dat
What is the filename of the BIM signal?
b.dat

The program will now print out the main menu and prompt:

Zero Adjustments	= 1
Amplitude Scaling	= 2
Normalization	= 3
Evaluating Contrast	= 4
Solving for Velocity	= 5
Insert a Velocity Jump	= 6
Graphics -- View Data	= 7
Undo Last Trace Alteration	= 8
Start All Over	= 9
Fully Automatic Reduction	= 10
Write Data To Disk	= 11
Done	= -1

Your selection is =

The steps from **1-6** should be executed in the order shown, and step **10** will do steps **1-5** with computer-identified input values (except for user-supplied values of the signal phase difference, the window material, if any, and the etalon length). After each step, the user can view the data files with selection **7**, or with a commercial software program if the data is output through selection **11.** If they look satisfactory, then proceed to the next step. Otherwise, selection **8** should be chosen and the data reduction step should be repeated. When done, enter **-1** and retrieve the particle velocity history from the file 'velocity.dat' in the program's directory (c:\visar).

To examine the current data set using the program's graphic routine, do the following:

Your selection is =
7
Pick data to plot:
BIM and Both Signals =1

Both Signals Only =2
Contrast =3
Velocity =4
Quit =0

Your Selection Is =
1
Plot Entire History (=0) or Selected Region (=1)?
0

The program will now create a new window which contains the graphical information. Its size and position can be controlled as with any window. When the user is finished viewing the graph,

Click and hold on the Window pull-down menu.

Drag down the choices and choose '1 Unit *' to return to the main program window.

Likewise, the user can return to the graphics window by doing the same procedure, except choosing '2 Graphic1' under the Window pull-down menu. In the Unit * window:

Activate the graphics window to view the graph...
Press RETURN to continue...
[Enter]
Choose one:

Select New Region =1
Back to Main Graphics Menu =2
Quit Graphics =0

Your Selection Is =
1
Input Starting Time =:
4.0e-6
Input Ending Time =:
8.0e-6
Activate the graphics window to view the graph...
Press RETURN to continue...

The program will automatically re-scale the y-axes for the plot. Viewing the graphics window is, again, done by using the Windows pull-down menu.

Press RETURN to continue...
[Enter]
Choose one:

Select New Region =1
Back to Main Graphics Menu =2

Quit Graphics =0

Your Selection Is =

0

The program will now display the main menu selections. To examine the current data set using SigmaPlot, do the following:

Output the signal data using selection **11** (see section L.).

Minimize the program window by clicking on the downward-pointing arrow in the upper right corner.

Find the SigmaPlot icon and double-click on this icon.

Click-and-hold on 'File' pull-down menu. Select 'Import Data' and change the active directory to 'c:\visar.'

The files with the current data set are 'signals.dat' and 'bim.dat' (as opposed to 'v1.dat', 'v2.dat' and 'b.dat' which are the original filenames).

Select, for example, 'signals.dat.'

The program will read the data and automatically deduce its format.

Choose 'Import' to place the data into columns 1, 2 and 3.

From the 'Graph' pull-down menu, select 'Create Graph'.

The automatic selection of 2D Cartesian / XY Pairs is correct.

Choose 'OK.' Then, choose 'Auto-Select' for setting the axes.

The program will automatically select column 1 for the x-axis and column 2 for the y-axis.

Choose 'OK' to accept the program's choice.

The plot will have automatically-scaled axes, but plot options can be set according to the user's preferences. Another plot can be made for the second trace by simply manually selecting columns 1 and 3 after calling the 'Create Plot for' option under the 'Plot' pull-down menu. In a similar manner, the 'bim.dat' file can be imported and plotted. Graphs of all three original-data files are shown in Figure 7. After viewing the data, return to the main program:

From the 'File' pull-down menu, choose 'Exit' and click 'No' to the warning message. Then find the main program icon, double-click on it, and proceed.

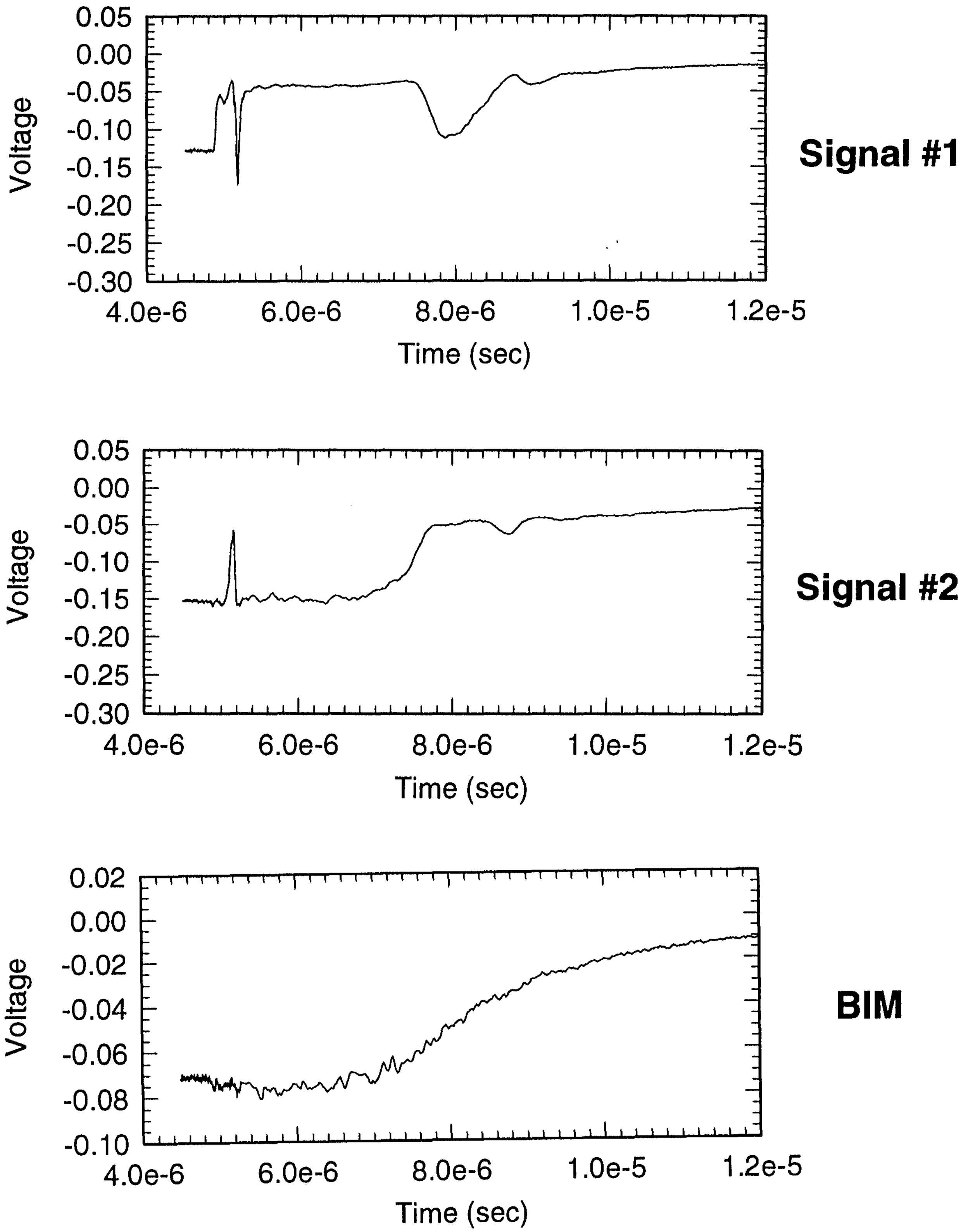


Figure 7 - Original data from a VISAR experiment.

D. Zero Adjustment

The first step is a zero adjustment. Here, constants are added to the data to make zero voltage equivalent to zero (interfering) light level. Experimentally, both signals are offset when data is taken because some amount of stray, noninterfering light always exists in the system. The program will find the voltage closest to zero for each trace and can automatically shift each trace by this amount. The option to manually zero the traces is also available. From the main menu:

```
Your selection is =
1
Choose Option:
        Automatically Zero Traces       =0
        Manually Zero Traces            =1
Your selection  is =
0
```

The program will indicate which voltage levels for each trace it found to be closest to zero. After viewing the traces, the user may wish to manually zero one or more traces. Simply choose **8** to undo the previous adjustment before entering **1** to re-do it manually. Figure 8 shows the traces after zero adjustments.

E. Amplitude Scaling

Next, effects of changes in beam intensity must be accounted for. The intensity of the light from the monitored surface can be affected by changes in reflectivity of this surface, defocusing of the beam sent to it, tilt during its motion, and self-lighting of strained material around it. Corrections can be made if the amplitudes of the BIM trace and the two VISAR records are all the same. Since a correction for this is very easy to do in the program, it is common practice in the experiment to minimize the amount of light sent to the beam-intensity PMT and thereby maximize the signal strengths. The next step, then, is to scale two of the three records so that the above condition holds. The options to automatically or manually scale the traces are available. If the automatic scaling option is chosen, the program will scale the traces so that the maximum intensities of each trace are the same (in this case, the maximum intensities occur at the most negative voltages). From the main menu:

```
Your selection is =
2
Choose Option:
        Automatically Scale Traces       = 0
        Manually Scale VISAR Trace #1    = 1
        Manually Scale VISAR Trace #2    = 2
        Manually Scale BIM Trace         = 3
        Done                             = -1
Your selection is =
```

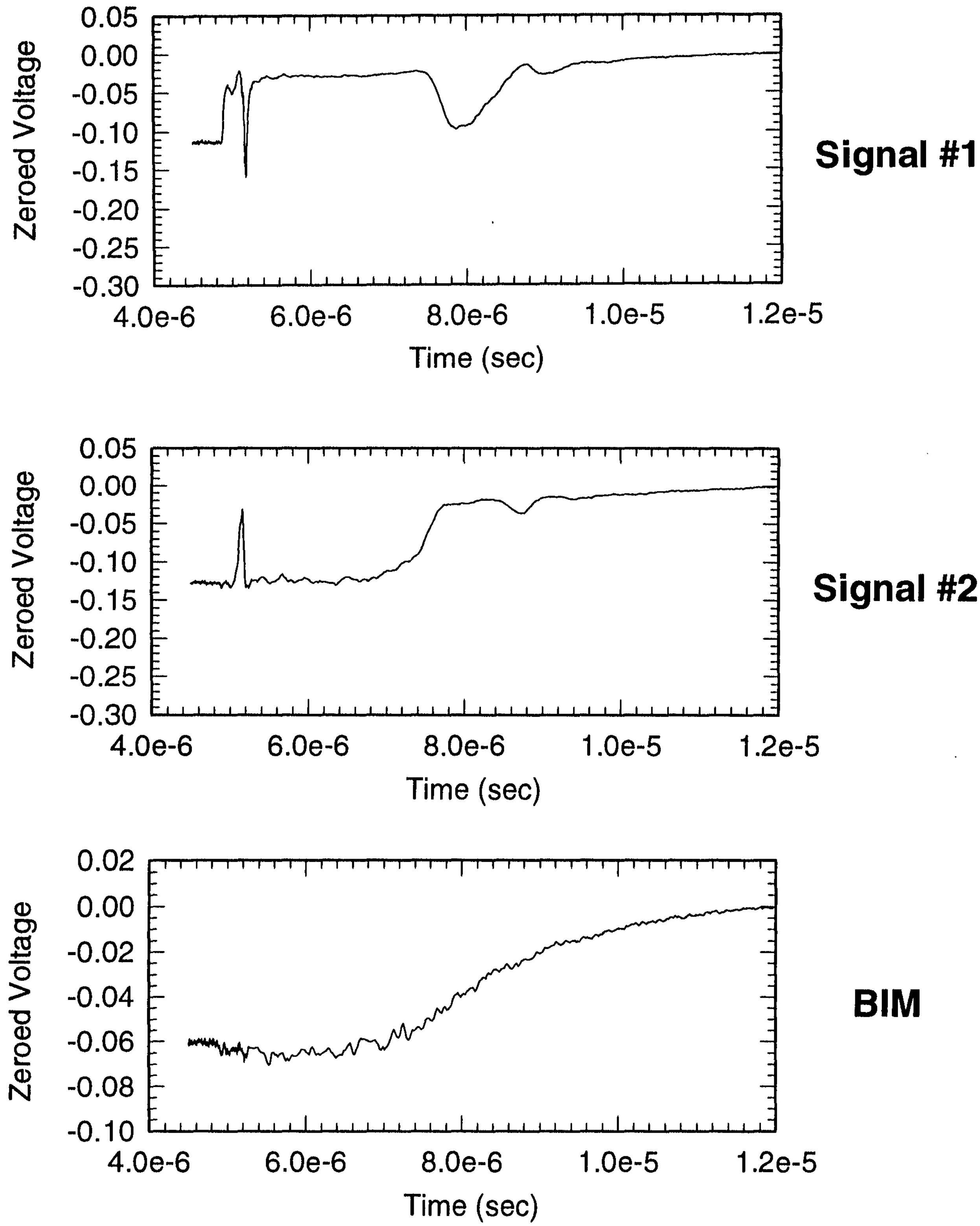


Figure 8 - Zero-adjusted records, where zero voltage corresponds to zero light.

0

Figure 9 shows the traces after their amplitudes have been scaled.

E. Normalization

The two traces should now be ready for normalization in order to eliminate the effects of beam-intensity variations. Choose **3** from the main menu to normalize the signals. Inside the program, the data is divided by the corresponding BIM amplitude at each time a reading was taken. The resulting records will then have values falling between 0 and -1, as shown in Figure 10.

F. Evaluating the Contrast

The normalized data can be represented in equation form as:

$$Y_1 = \tfrac{1}{2}\left[-1 + C\sin(2\pi F(t) + \theta_o)\right] \tag{3.1}$$

$$Y_2 = \tfrac{1}{2}\left[-1 + C\sin(2\pi F(t) + \theta_o + \varphi)\right] \tag{3.2}$$

Here, *C(t)* is the 'contrast' function, *F(t)* is the fringe count, θ_o is the initial phase at $F(t) = 0$, and φ is the phase difference between the two VISAR signals (determined in Section B). They represent two equations with two unknowns, *C(t)* and *F(t)*. The fact that *C* changes during recording time is due to the frequency response of the entire VISAR instrumentation. These equations may be solved for *C(t)*, giving:

$$C = \sqrt{(2Y_1 - 1)^2 + \left[\frac{2Y_2 - 1 - (2Y_1 - 1)\cos\varphi}{\sin\varphi}\right]^2}\ . \tag{3.3}$$

Once $C(t)$ is known, eqn. (3.1) and eqn. (3.2) can be inverted to obtain the function *F(t)* (see Section G). To calculate the contrast values for the sample traces:

Your selection is =
4
The phase diff between the signals (deg)?
-100.

The user can examine the contrast function in SigmaPlot. It is stored under the filename 'c.dat' and shown in Figure 11. The fact that the contrast is above 1.0 in the early part of the record is due to a need for manually scaling the BIM trace; however, since the contrast is divided out before calculating the particle velocity, the final result will be the same.

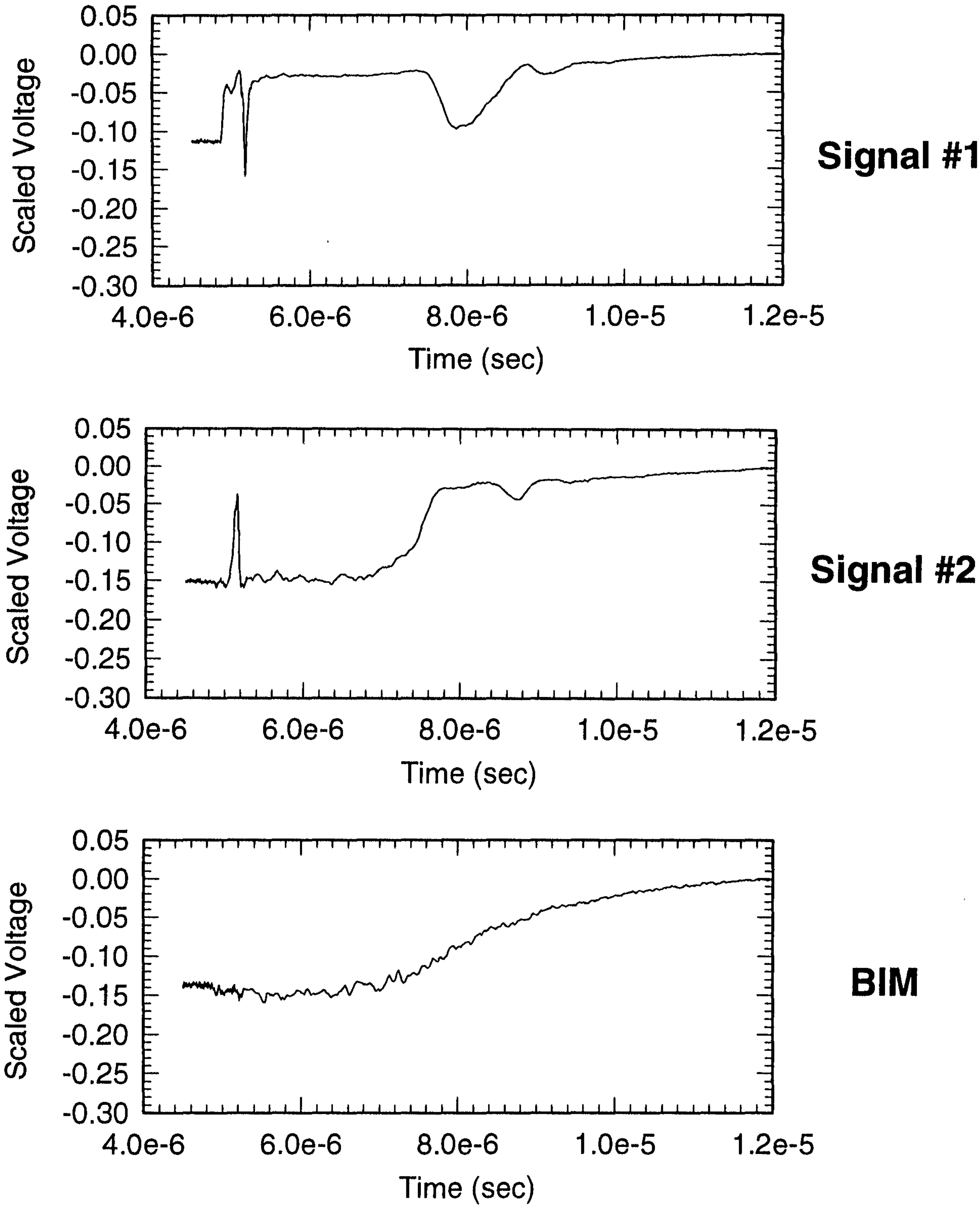


Figure 9 - Scaled records so that all traces have essentially the same amplitude.

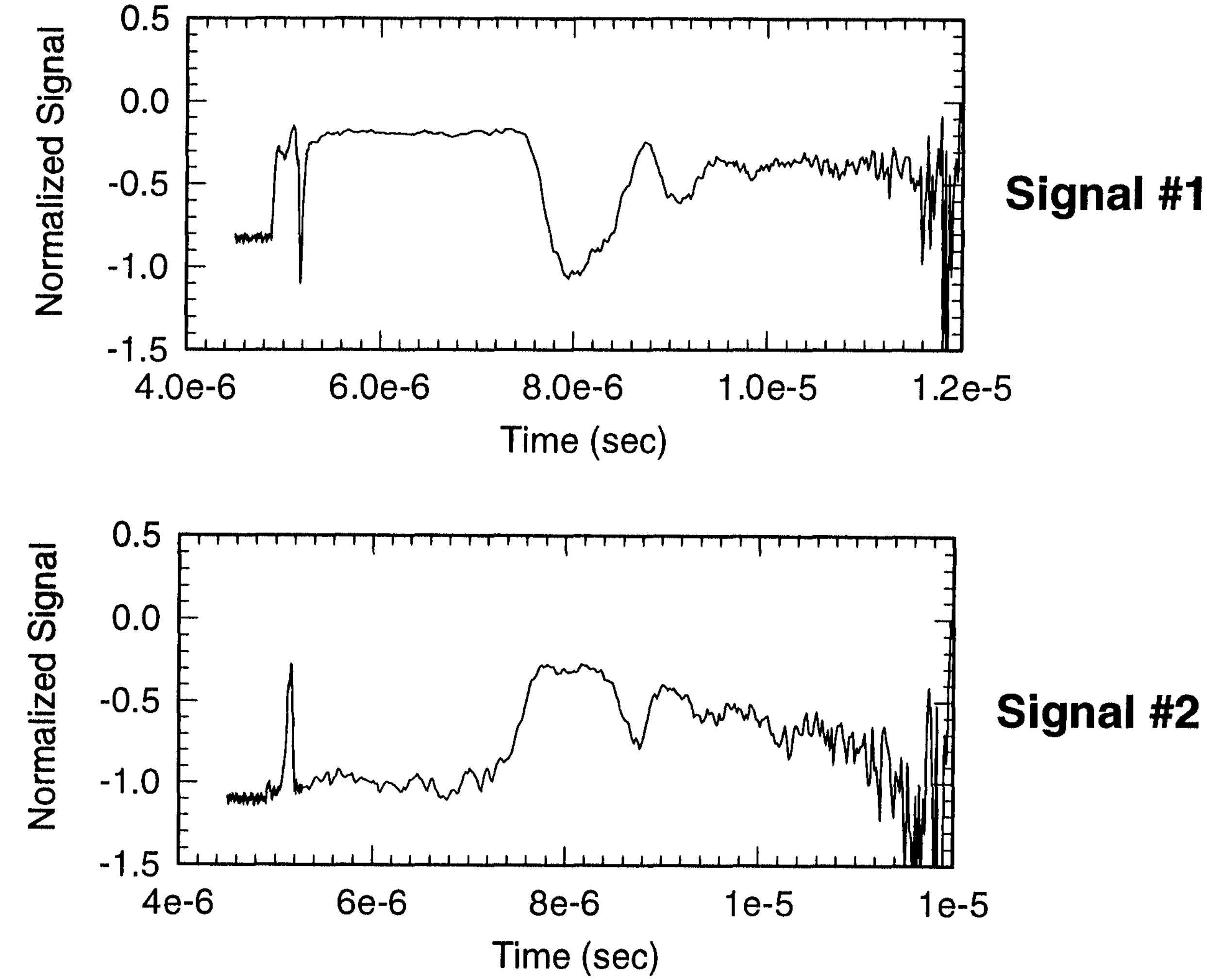


Figure 10 - Two traces after correcting for BIM variations ('normalization').

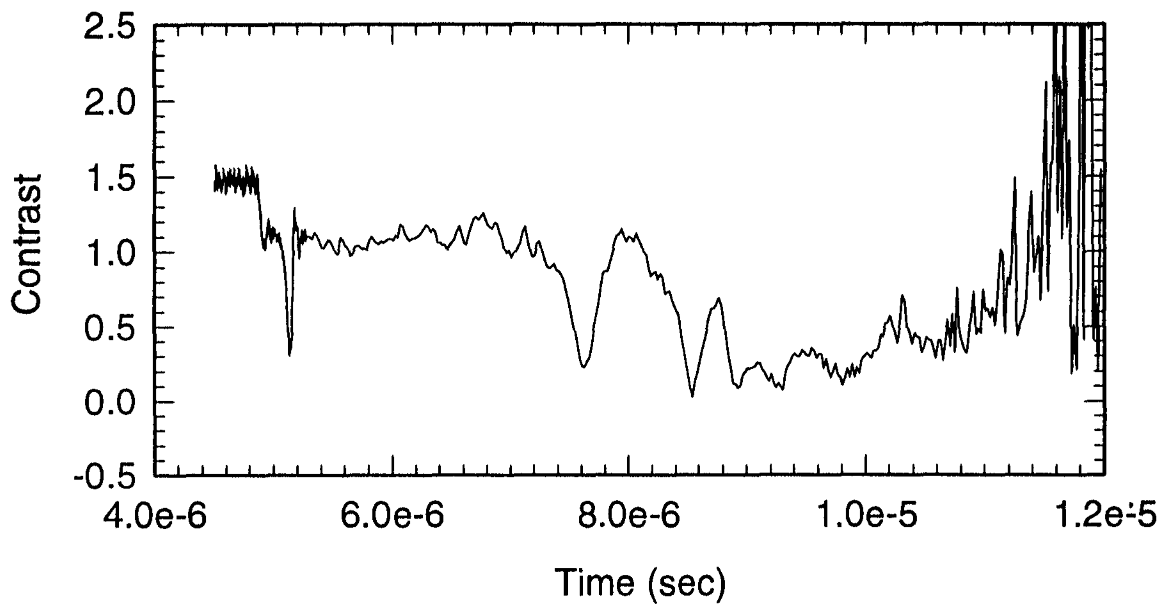


Figure 11 - Calculated contrast function from normalized records.

G. Calculation of Particle Velocity

The function *F(t)* is calculated by inverting either eqn. (3.1) or (3.2). Since it involves the arcsine of some number, the computer will return a solution between $-\pi/2$ and $\pi/2$. There will also be a valid solution in the range $\pi/2$ to $3\pi/2$. The program uses the experimental value of the other trace to determine which solution is correct. Phase angles, θ, are calculated for each successive point in the data, starting with θ_o. Each increase or decrease in phase angle is added to *F(t)*, giving the fringe count for every point.

Each value of *F(t)* is substituted into (2.23) to obtain $u(t-\tau/2)$. If a window material is used, then $\Delta\nu/\nu_o$ depends on u and an iteration process is carried out in the program which determines $u(t-\tau/2)$. Since $\Delta\nu/\nu_o$ is a slowly varying function of u, quick convergence to the desired $u(t-\tau/2)$ is assured. To execute the velocity calculation:

```
Your selection is =
5
What was the window material?
        PMMA                      = 1
        Fused Silica              = 2
        Z-Cut Sapphire            = 3
        Lithium Fluoride          = 4
        NONE (Free-Surface Shot)  = 5
Your selection is =
5
The total length of etalon matl.(inches)?
6.
```

The resulting velocity record can be viewed using selection **7** of the main menu. Another way to view the data is to choose selection **11** and store the velocity record (file 'velocity.dat' in directory 'c:\visar'). It can then be viewed using SigmaPlot. A COPS code calculation, 'cops.dat', for this experiment is included with the sample files for comparison. Figure 12 shows the resulting particle velocity calculation along with the COPS prediction.

H. Insertion of Velocity 'Jumps'

In some instances, the acceleration or deceleration of the monitored surface is so large that the instrumentation cannot resolve it. In such cases, one or more fringes may not be present in the records. Often, one knows roughly what velocity to expect and since the velocity corresponding to one fringe is usually large, determination of exactly how many fringes were lost is usually possible. The main program allows for the addition of lost fringes anywhere in the velocity profile. For example, from the main menu:

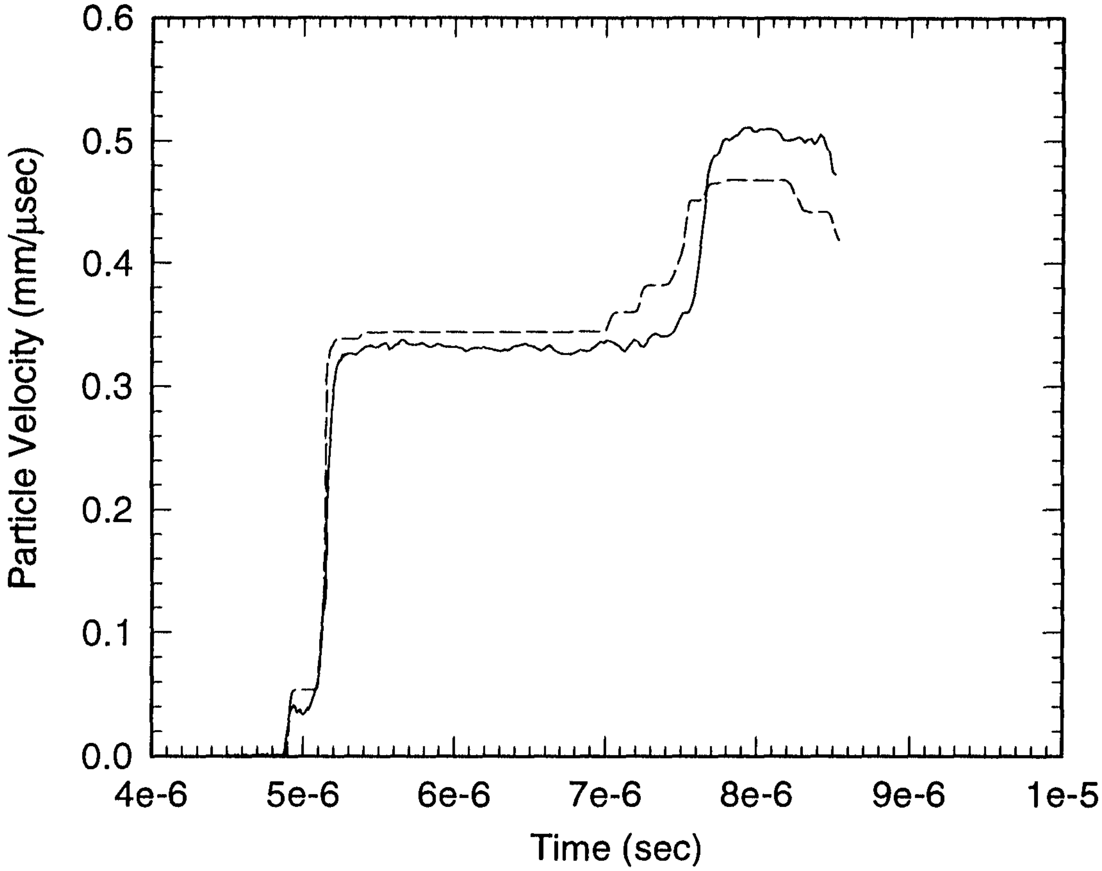


Figure 12 - Calculated particle-velocity history and COPS prediction.

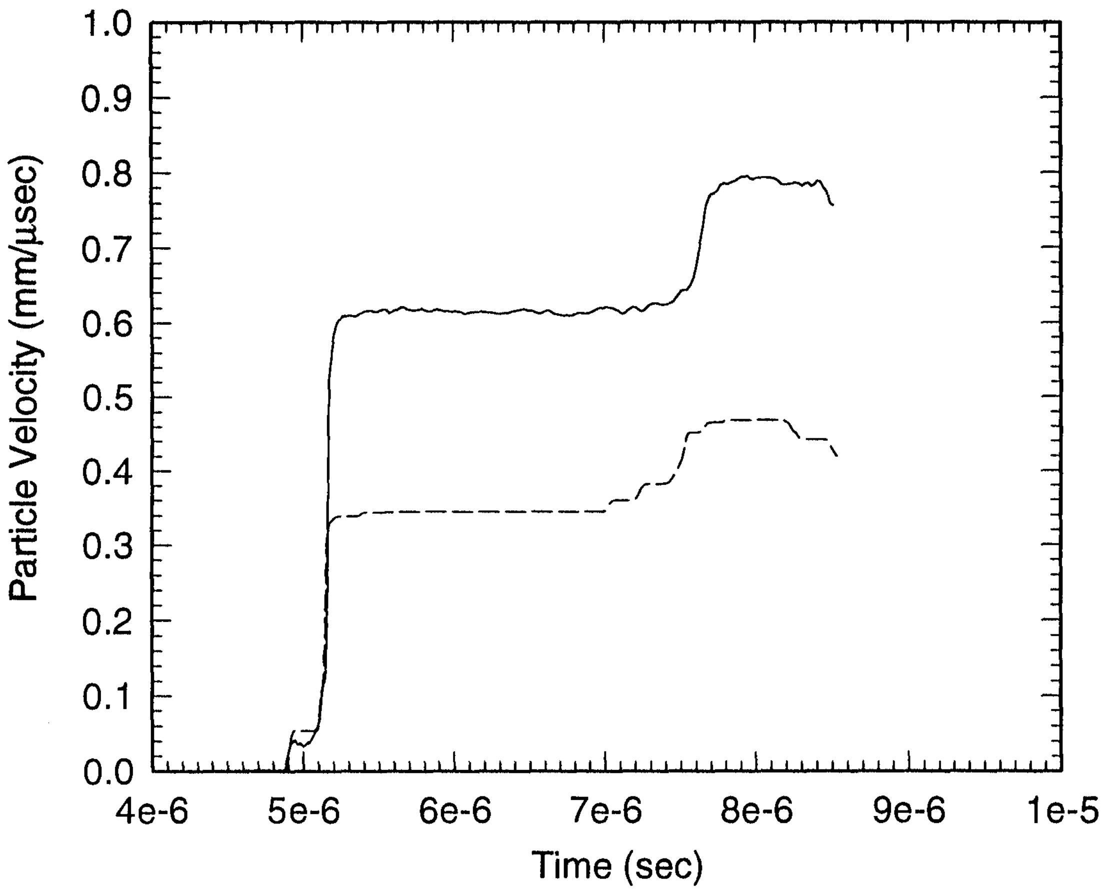


Figure 13 - Calculated particle-velocity after addition of one fringe.

Your selection is =
6
What is the starting time of the jump (sec)?
5.16e-6
What is the ending time of the jump (sec)?
5.17e-6
How many fringes to insert?
1
Now, to look at this new velocity record,
Activate the graphics window to view the graph...
Press RETURN to continue...

If the graphics window does not automatically come up, click on the Windows pull-down menu and release on '2Graphic1'. The data at this point will look like Figure 13. Obviously, the addition of one fringe causes an incorrect result. Therefore, there were no lost fringes in this experiment.

[Enter]
Choose one:
    Select New Region =1
    Quit Graphics =2
Your Selection Is =
2
Choose an option:
    Accept the Change = 1
    Try Again = 2
    Quit (Leave Data As Before) = 3
Your selection is =

Choice **3** should be made from the list above.

I. Graphics

A graphics routine is included for convenient examination of the data without having to exit the program. It allows the user to view different types of data and also to zoom in (in time) on parts of the records. The y-scale is automatically chosen by the program. For example, if the user wanted to examine the raw data at the beginning of the program, the following sequence would be used:

Your selection is =
7
Pick data to plot:
    BIM and Both Signals =1
    Both Signals Only =2
    Contrast =3

Velocity =4
Quit =0
Your Selection Is =
1
Plot Entire History (=0) or Selected Region (=1)?
0

The displayed graphics window will look like Figure 14. To toggle between the graphics window and the main menu window, use the Window pull-down menu and select between '1 Unit*' and '2 Graphic1'. Zooming in on a record can be done as follows (from the main program window):

Activate the graphics window to view the graph...
Press RETURN to continue...
[Enter]
Choose one:
Select New Region =1
Back to Main Graphics Menu =2
Quit Graphics =0
Your Selection Is =
1
Input Starting Time =:
4.e-6
Input Ending Time =:
8.e-6
Activate the graphics window to view the graph...
Press RETURN to continue...

Now, the graphics window display will resemble Figure 15. Going back to the main program, the user can view other data (like the contrast or the velocity) or go back to the main menu. If more flexibility is needed in viewing the data, it can be output using selection **11** (see section L), and a commercial graphics software program may be used.

J. Undoing a Step or Starting All Over

While trying to reduce data manually, it may be necessary to try a routine several times. Should this be the case, executing step **8** (Undo Last Trace Alteration) before repeating a given step is essential. This will reset the data arrays to what they were before executing the errant step. If irrecoverable changes are made to the data, step **9** (Start All Over) is available and will reset the data arrays to what they were before any data reduction took place.

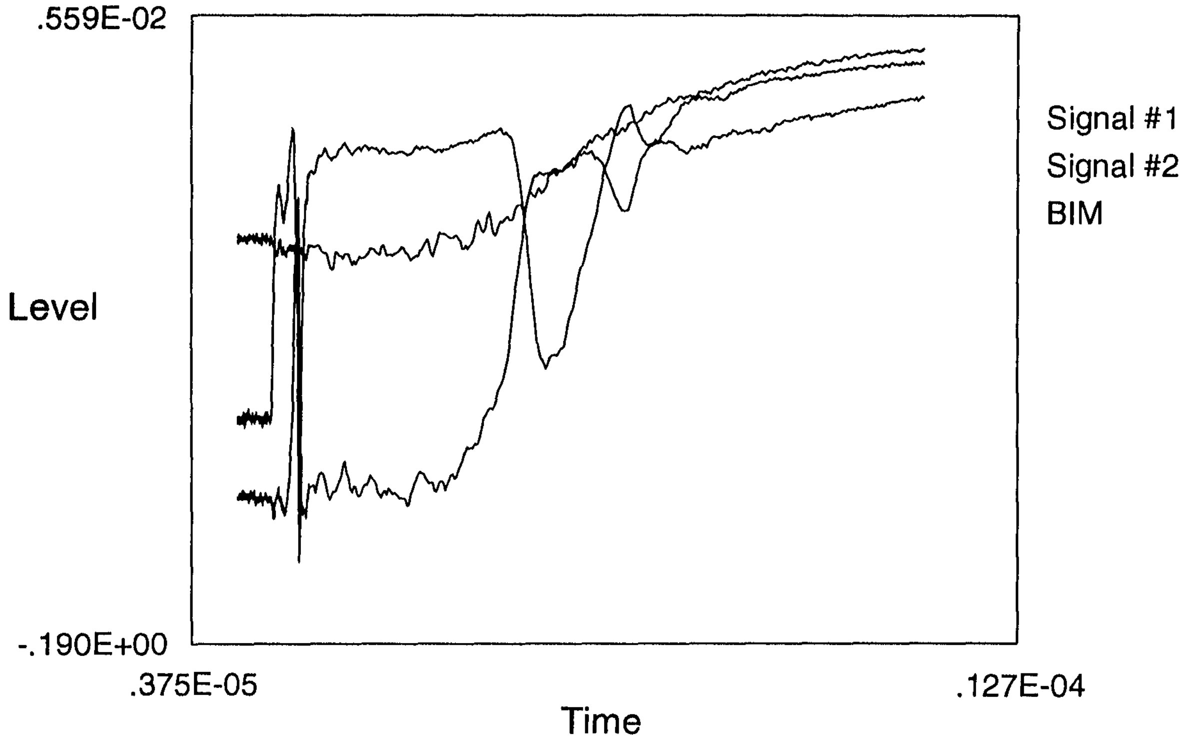


Figure 14 - Graphics window representation of the original data.

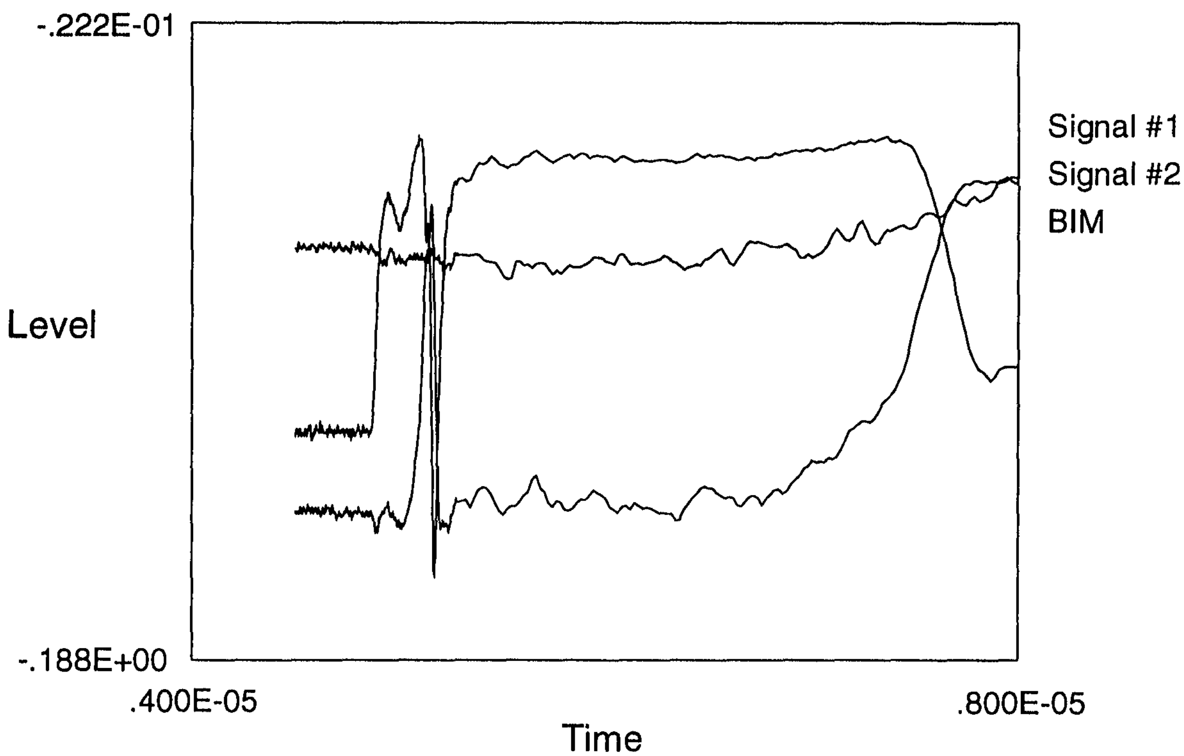


Figure 15 - New (zoomed in) region of the original data plotted using the graphics routine.

K. Fully Automatic Data Reduction

A good first step (which works in the case of the sample data provided here) is often to choose **10** (Fully Automatic Reduction), and then view the results to see if they look reasonable. If the data is clean and signal noise is low, this step should work and takes no time to execute, other than entering three pieces of data necessary for the program. For the sample data, it can be executed any time by starting in the main menu and doing the following:

> Your selection is =
> **9**

The program will reset the data and re-display the main menu.

> Your selection is =
> **10**
> The phase diff between the signals (deg)?
> **-100.**
> What was the window material?
>
> | | |
> |---|---|
> | PMMA | = 1 |
> | Fused Silica | = 2 |
> | Z-Cut Sapphire | = 3 |
> | Lithium Fluoride | = 4 |
> | NONE (Free-Surface Shot) | = 5 |
>
> Your selection is =
> **5**
> The total length of etalon matl.(inches)?
> **6.**

The program will carry out the steps **1-5** automatically, giving the velocity history shown in Figure 12.

L. Writing Data to Disks

If the user wishes to store data onto disks (for help in data reduction outside of the main program or for plotting the results) menu selection **11** is available.

> Your selection is =
> **11**
> Choose:
>
> | | |
> |---|---|
> | Write Signal Data To Disk | = 1 |
> | Write Velocity Data To Disk | = 2 |
> | Write Contrast Data To Disk | = 3 |
> | Write Fringe Data To Disk | = 4 |
> | Return To Main Menu | = -1 |

Your selection is=
2

The program will then store the velocity data into the file 'velocity.dat' inside the working directory 'c:\visar'. For the signal data, the file is 'signals.dat' and for the BIM, the file is 'bim.dat'. The contrast data is stored in 'contrast.dat' and the fringe count data is in 'fringec.dat'.

V. References

[1] Barker, L.M. and Hollenbach, R.E., J. Appl. Phys. **43**, 4669 (1972).

[2] Barker, L.M. and Hollenbach, R.E., J. Appl. Phys. **41**, 4208 (1970).

[3] Clifton, R.J., J. Appl. Phys. **41**, 5335 (1970).

[4] Barker, L.M., Exp. Mech. **12**, 40 (1972).

[5] L.M. Barker, in *Behavior of Dense Media Under High Dynamic Pressures* (Gordon and Breach, New York, 1968), p. 483.

[6] Wise, J.L., Shock Waves in Condensed Matter - 1989, 441 (1989).

[7] Zwick, H.H. and Shephard, G.G., Appl. Opt. **10**, 2569 (1971).

[8] A discussion of lens delay legs is beyond the scope of this work.

[9] Barker, L.M. and Schuler, K.W., J. Appl. Phys. **45**, 3692 (1974).

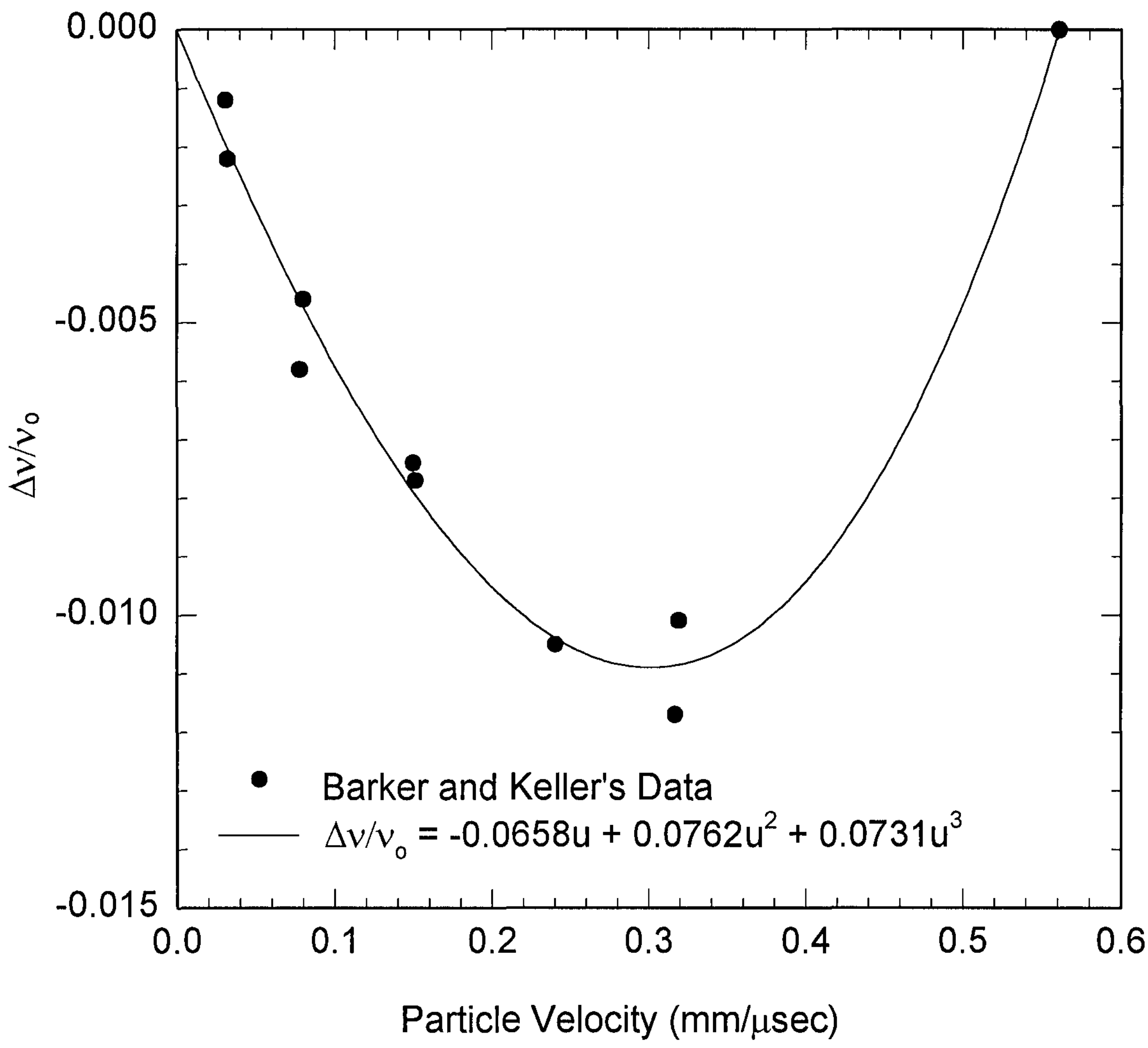
[10] Barker, L.M., personal communication.

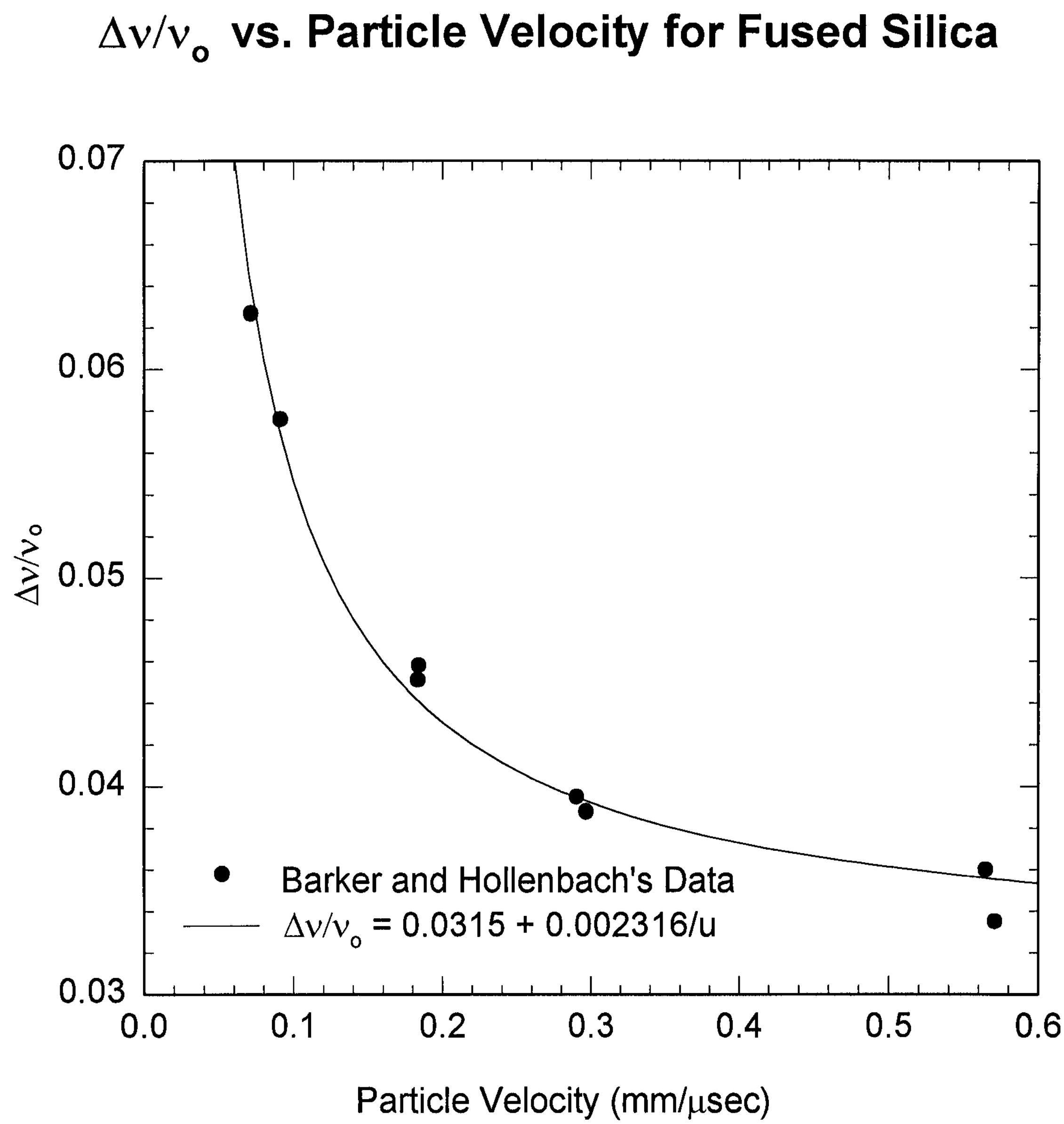
[11] Barker, L.M., Sandia Report SAND88-2788, (1988).

Appendix A - Window Material Correction Factors

The following plots indicate the frequency correction factors, $\Delta\nu / \nu_o$, used in the SDC VISAR data reduction program. Data were taken from Barker and Hollenbach [2] for PMMA, fused silica and sapphire and from Wise [6] for lithium fluoride. In each case, curve-fitting was accomplished and the results are indicated on the plots. In the program, a warning is given whenever the particle velocity extends beyond the calibrated data in these figures.

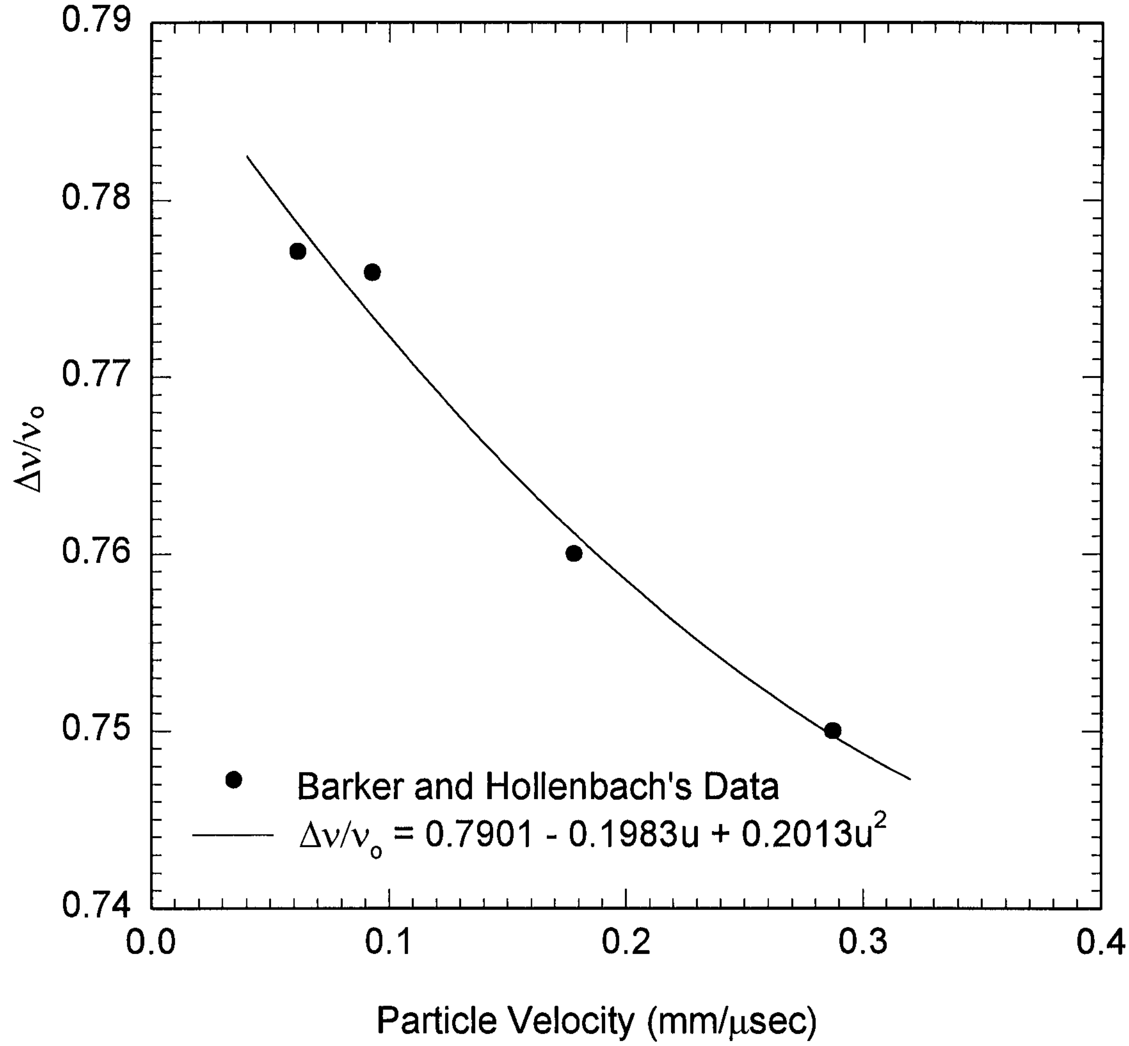
$\Delta\nu/\nu_o$ vs. Particle Velocity for PMMA



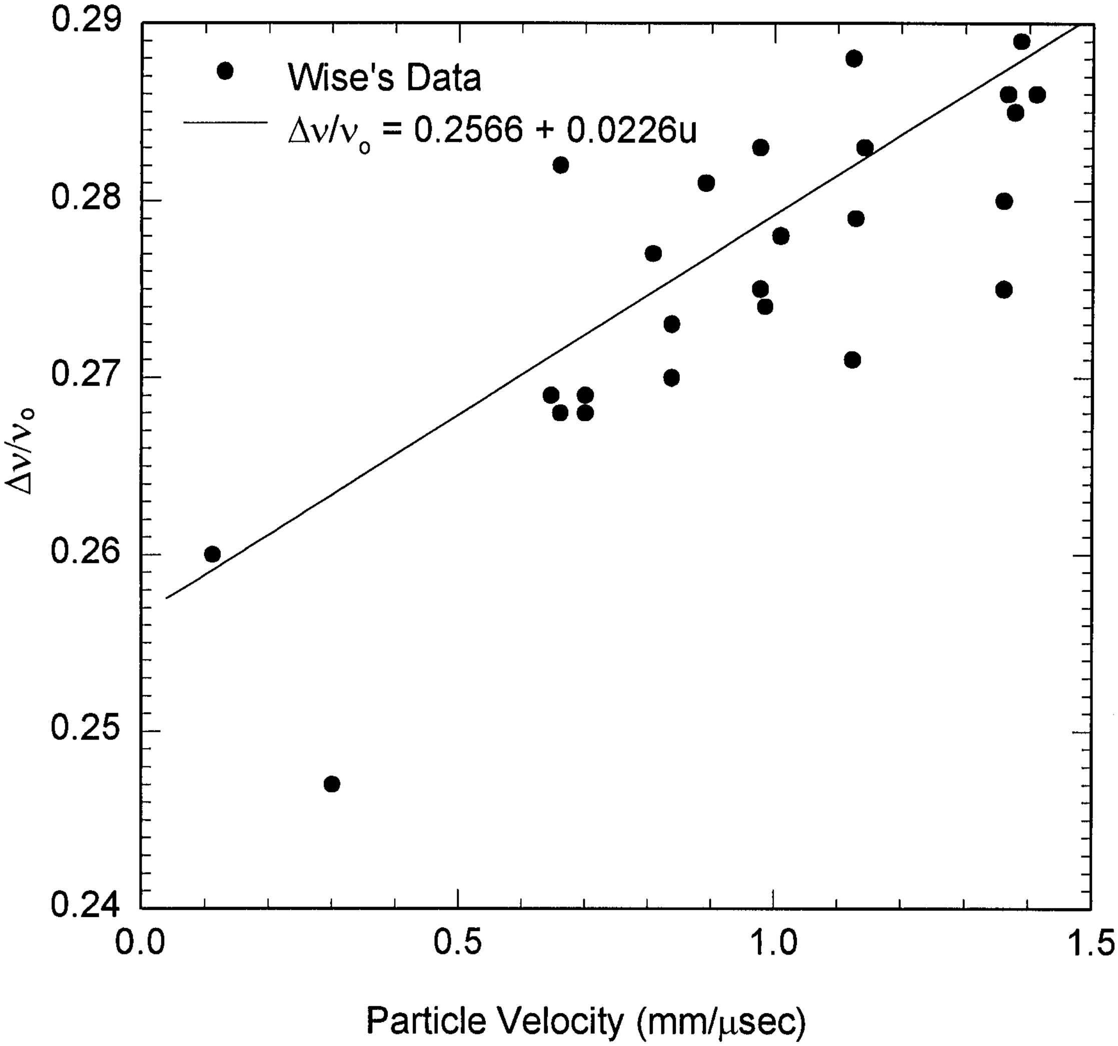


Δν/ν₀ vs. Particle Velocity for Fused Silica
0.07
0.06
0.05
0.04
0.03
Δν/ν₀
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Particle Velocity (mm/μsec)
Barker and Hollenbach's Data
Δν/ν₀ = 0.0315 + 0.002316/u

$\Delta\nu/\nu_o$ vs. Particle Velocity for Z-Cut Sapphire



$\Delta\nu/\nu_o$ vs. Particle Velocity for LiF



Appendix B - Program List

The following is simply a list of the SDC data reduction program. For ease of use, the program is put into a windows environment and any data visualization software can be chosen to view the data after executing step **11** of the main menu.

```fortran
c    program sdcvis
c
c    Latest Revision: 10/29/94
c
c    This program allows the user to carry out all the steps
c    necessary for VISAR data reduction from the WSU Shock
c    Dynamics Center VISAR.  Most steps follow those developed
c    by L. Barker in his Sandia Report on VISAR88.  Specifically:
c
c        I. Zero Adjustments - Add constants to data to make
c                        zero deflection on the scope
c                        record correspond to zero light.
c
c        II. Amplitude Scaling - Assures that both traces have
c                         equal fringe min and max and that
c                         the BIM has twice the average of
c                         each.
c
c        III. Normalization - Eliminates the effects of BIM variations
c                       by dividing the data by the corresponding
c                       BIM amplitude at each time a reading
c                       was taken.
c
c        IV. Determination of Contrast - Knowing the amplitudes of
the
c                              two traces and the phase
c                              difference between them, the
c                              contrast can be solved for.
c
c        V. Determination of Velocity - The fringe count for a given
c                             record is calculated and the
c                             velocity is derived using the
c                             fringe constant and window
c                             corrrections, if necessary.
c
c        VI. Insertion of Vel. Jumps - If a velocity change is too
c                             large for the VISAR to resolve,
c                             lost fringes can be added
c                             manually.
c
c
c    Files:
c
c        vis1    - Original data trace #1
c        vis2    - Original data trace #2
c        bim     - Original BIM trace
c
c
c    Subroutines:
c
c        zero        - Performs zero adjustments as in I. above.
c        scale       - Performs amplitude scaling as in II.
c        normal      - Performs normalization as in III.
c        contrast    - Determines the contrast function as in IV.
c        vel         - Calculates the particle-velocity history as in V.
c        jump        - Inserts integer number of fringes into velocity
c                      record.
c        window      - Calculates window correction factor during
c                      velocity calculation.
c        storeold    - Moves the *.tmp files to the *.old files.
c        shift       - Moves the *.dat files to the *.tmp files.
c        undo        - Returns the data files to what they were prior
c                      to the last data reduction step.
c        reset       - Returns the data files to what they were at
c                      the beginning of the data reduction process.
c        question    - Questions the user about whether to zero a given
c                      trace manually or automatically.
c        writedata   - Writes the current data (signal, bimdat) to disk
c        writevel    - Writes the velocity record (vel) to disk
c        writecontrast - Writes the contrast data (ctrast) to disk
c        writefringe - Writes the fringe count data (f) to disk
c        graphics    - The graphics routine called by the main menu
c        plot        - Plots data into the graphics window
c                      and adds labels
c        srange      - Finds the ranges for plotting user-selected parts
c                      of a data array
c        erange      - Finds the ranges for plotting an entire data array
c
c
c    Variables are defined at the beginning of each subroutine.
c
c    Variables in this main program:
c
c        signal  - The data (both signals) after the last reduction step
c        bimdat  - The BIM after the last reduction step
c        tmpsig  - The data prior to the last reduction step
c        tmpbim  - The BIM prior to the last reduction step
c        oldsig  - The data prior to the last two reduction steps
c        oldbim  - The BIM prior to the last two reduction steps
c        ctrast  - The contrast data array
c        f       - The fringe count data array
c        vel     - The calculated particle velocity array
c        ftry    - Temp. stored fringe count after inserting vel. jump
c        vtry    - Temp. stored velocity after insertion of vel. jump
c        iselect - Stores the selection of a main data reduction step.
c        phi     - User-supplied phase shift between the traces
c        frconst - The fringe constant (without a window correction)
c        nwin    - Stores the type of window used in the experiment
c        error   - Finds whether vis1,vis2,bim exist
c        npsig   - Number of data points in the VISAR traces
c        npbim   - Number of data points in the BIM trace
c
c    First, include the Microsoft Fortran graphics files...
c
      include 'fgraph.fi'
      include 'fgraph.fd'
c
      double precision signal(10001,3),tmpsig(10001,3)
      double precision bimdat(10001,2),tmpbim(10001,2)
      double precision vel(10001,2),f(10001,2),ctrast(10001)
      double precision oldsig(10001,3),oldbim(10001,2)
      double precision ftry(10001,2),vtry(10001,2)
      double precision a,b,c,d,phi,frconst
      character*40 vis1,vis2,bim
      integer error,npsig,npbim
      common /list1/ signal, npsig
      common /list2/ tmpsig
      common /list3/ bimdat, npbim
      common /list4/ tmpbim
      common /list5/ vel
      common /list6/ f
      common /list7/ ctrast
      common /list8/ oldsig, oldbim
      common /list9/ ftry,vtry
c
      write(*,10)
 10   format(1x)
c
c
c     Print out logo first...
c
c
      write(*,12)
 12   format(/
     c /11x,'*********************************************'
     c /11x,'*                                           *'
     c /11x,'*      WSU Shock Dynamics Lab VISAR         *'
     c /11x,'*                                           *'
     c /11x,'*        Data Reduction Program             *'
     c /11x,'*                                           *'
     c /11x,'*        (Written by G. Raiser)             *'
     c /11x,'*                                           *'
     c /11x,'*        Last Revision: 10/29/94            *'
     c /11x,'*                                           *'
     c /11x,'*********************************************'
     c ///)
      write(*,10)
      write(*,10)
 1000 write(*,20)
 20   format(1x,'What is the filename of VISAR signal #1 ? ')
      read(*,30)vis1
      open(unit=1,file=vis1,status='old',iostat=error)
      if (error.gt.0) then
            write(*,10)
            write(6,*) 'Unable to open file, please reenter filename'
            write(*,10)
```

```
            go to 1000
      endif
30    format(a40)
      write(*,10)
2000  write(*,40)
40    format(1x,'What is the filename of VISAR signal #2 ? ')
      read(*,30)vis2
      open(unit=2,file=vis2,status='old',iostat=error)
      if (error.gt.0) then
            write(*,10)
            write(6,*) 'Unable to open file, please reenter filename'
            write(*,10)
            goto 2000
      endif
      write(*,10)
3000  write(*,50)
50    format(1x,'What is the filename of the BIM signal ? ')
      read(*,30)bim
      open(unit=3,file=bim,status='old',iostat=error)
      if (error.gt.0) then
            write(*,10)
            write(6,*) 'Unable to open file, please reenter filename'
            write(*,10)
            goto 3000
      endif
c
c     Transfer the file data into the variables...
c
      i = 1
      read(1,*,end=93) a, b
      read(2,*,end=93) c, d
5     tmpsig(i,1) = a
            tmpsig(i,2) = b
            tmpsig(i,3) = d
            signal(i,1) = a
            signal(i,2) = b
            signal(i,3) = d
            oldsig(i,1) = a
            i = i + 1
            read(1,*,end=93) a, b
            read(2,*,end=93) c, d
      go to 5
93    write(*,10)
      npsig = i - 1
      i = 1
      read(3,*,end=94) a, b
3100  bimdat(i,1) = a
            bimdat(i,2) = b
            tmpbim(i,1) = a
            tmpbim(i,2) = b
            oldbim(i,2) = b
            i = i + 1
            read(3,*,end=94) a, b
      goto 3100
94    write(*,10)
      npbim = i-1
      close(1)
      close(2)
      close(3)
c
      frconst=0.0
55    write(*,10)
      write(*,10)
      write(*,60)
60    format(//1x,'Main Menu Selections:'
     c     //1x,'   Zero Adjustments           = 1'
     c     /1x,'    Amplitude Scaling          = 2'
     c     /1x,'    Normalization              = 3'
     c     /1x,'    Evaluating Contrast        = 4'
     c     /1x,'    Solving for Velocity       = 5'
     c     /1x,'    Insert a Velocity Jump     = 6'
     c     /1x,'    Graphics -- View Data      = 7'
     c     /1x,'    Undo Last Trace Alteration = 8'
     c     /1x,'    Start All Over             = 9'
     c     /1x,'    Fully Automatic Reduction  = 10'
     c     /1x,'    Write Data To Disk         = 11'
     c     /1x,'    Done                       = -1'
     c     //)
```

```
      write(*,10)
      write(*,70)
70    format(1x,'Your selection is = ')
      read(*,80)iselect
80    format(i2)
      if (iselect.eq.1) then
         call storeold(npsig,npbim)
              call shift
         call zero(iselect)
      endif
      if (iselect.eq.2) then
         call storeold(npsig,npbim)
              call shift
         call scale(iselect)
      endif
      if (iselect.eq.3) then
         call storeold(npsig,npbim)
              call shift
              call normal
      endif
      if (iselect.eq.4) call contrast(phi)
      if (iselect.eq.5) call velocity(frconst,nwin,phi)
      if (iselect.eq.6) then
              if (frconst.eq.0.0) then
                     write(*,100)
100           format(1x,'Calculate the velocity history first!')
                     go to 55
              else
           call jump(frconst,nwin,npsig)
              endif
      endif
      if (iselect.eq.7) call graphics
      if (iselect.eq.8) call undo(npsig,npbim)
      if (iselect.eq.9) call reset(vis1,vis2,bim)
      if (iselect.eq.10) then
         call storeold(npsig,npbim)
         call shift
         call zero(iselect)
         call storeold(npsig,npbim)
              call shift
         call scale(iselect)
         call storeold(npsig,npbim)
              call shift
         call normal
         call contrast(phi)
         call velocity(frconst,nwin,phi)
      endif
      if (iselect.eq.11) then
115      write(*,10)
         write(*,120)
120      format(//1x,'Choose:'
     c   //1x,'   Write Signal Data To Disk   = 1'
     c   /1x,'    Write Velocity Data To Disk = 2'
     c   /1x,'    Write Contrast Data To Disk = 3'
     c   /1x,'    Write Fringe Data To Disk   = 4'
     c   /1x,'    Return to Main Menu         = -1'
     c   //)
         write(*,10)
         write(*,130)
130      format(1x,'Your selection is = ')
         read(*,80)iselect
         if (iselect.eq.1) call writedata
         if (iselect.eq.2) call writevel(npsig)
         if (iselect.eq.3) call writecontrast(npsig)
         if (iselect.eq.4) call writefringe(npsig)
         go to 55
      endif
      if (iselect.eq.-1) go to 999
      go to 55
999   stop
      end
c
      subroutine shift
c
c     subroutine to shift signal-->tmpsig and bimdat-->tmpbim
c
      double precision signal(10001,3),tmpsig(10001,3)
      double precision bimdat(10001,2),tmpbim(10001,2)
```

```
      integer npsig,npbim
      common /list1/ signal, npsig
      common /list2/ tmpsig
      common /list3/ bimdat, npbim
      common /list4/ tmpbim
c
      do k = 1,npsig
            tmpsig(k,2) = signal(k,2)
            tmpsig(k,3) = signal(k,3)
      end do
      do k = 1,npbim
            tmpbim(k,1) = bimdat(k,1)
            tmpbim(k,2) = bimdat(k,2)
      end do
      return
      end
c
      subroutine zero(iselect)
c
c      To zero the VISAR records before amplitude scaling.
c
c      Variables:
c          vhigh   - the highest voltage found for a given trace
c          iselect - the main menu selection
c
      double precision signal(10001,3),tmpsig(10001,3)
      double precision bimdat(10001,2),tmpbim(10001,2)
      double precision vhigh1,vhigh2,vhigh3
      integer npsig,npbim,iselect,ichoice
      common /list1/ signal, npsig
      common /list2/ tmpsig
      common /list3/ bimdat, npbim
      common /list4/ tmpbim
c
 10   format(1x)
 11   write(*,10)
      if (iselect.eq.10) then
        ichoice=0
        go to 19
      endif
      write(*,12)
 12   format(//1x,'Choose Option:'
     c      //1x,'  Automatically Zero Traces      = 0'
     c      /1x,'   Manually Zero Traces           = 1'
     c      //)
      write(*,10)
      write(*,13)
 13   format(1x,'Your selection is = ')
      read(*,14)ichoice
 14   format(i2)
      write(*,10)
      if (ichoice.eq.0.or.ichoice.eq.1) go to 19
      go to 11
c
c       The first VISAR trace
c
 19   vhigh1=-1000.0d0
      vhigh2=-1000.0d0
      vhigh3=-1000.0d0
c
c       First, find the highest voltage (weakest signal)
c
      do k = 1, npsig
       if (tmpsig(k,2).gt.vhigh1) vhigh1=tmpsig(k,2)
       if (tmpsig(k,3).gt.vhigh2) vhigh2=tmpsig(k,3)
      end do
      do k = 1, npbim
       if (tmpbim(k,2).gt.vhigh3) vhigh3=tmpbim(k,2)
      end do
      if (iselect.eq.10) go to 30
      call question(vhigh1,vhigh2,vhigh3,ichoice)
 30   do k = 1, npsig
            vnew1=tmpsig(k,2)-vhigh1
            vnew2=tmpsig(k,3)-vhigh2
            if (vnew1.gt.0.0d0) vnew1=0.0d0
            if (vnew2.gt.0.0d0) vnew2=0.0d0
            signal(k,2)=vnew1
            signal(k,3)=vnew2
      end do
      do k = 1, npbim
            vnew3=tmpbim(k,2)-vhigh3
            if (vnew3.gt.0.0d0) vnew3=0.0d0
            bimdat(k,2)=vnew3
      end do
      return
      end
c
      subroutine question(vhigh1,vhigh2,vhigh3,ichoice)
c
c       To allow the user to override the automatic
c       zero option.
c
      implicit double precision (a-h,o-z)
c
      write(*,10)
 10   format(1x)
      write(*,20)vhigh1
 20   format(1x,'Trace 1 Scanned Highest Voltage = ',f18.14)
      write(*,10)
      write(*,21)vhigh2
 21   format(1x,'Trace 2 Scanned Highest Voltage = ',f18.14)
      write(*,10)
      write(*,22)vhigh3
 22   format(1x,'BIM Scanned Highest Voltage = ',f18.14)
      write(*,10)
      if (ichoice.eq.0) go to 120
      write(*,80)
 80   format(1x,'Trace 1 Zeroed manually (=1) or automatically (=0)? ')
      read(*,90)nans
 90   format(i3)
      if (nans.eq.1) then
            write(*,10)
            write(*,100)
 100    format(1x,'What is the voltage (now to be zeroed)? ')
            read(*,110)vans
 110    format(f20.15)
            vhigh1=vans
      endif
      write(*,81)
 81   format(1x,'Trace 2 Zeroed manually (=1) or automatically (=0)? ')
      read(*,90)nans
      if (nans.eq.1) then
            write(*,10)
        write(*,100)
        read(*,110)vans
        vhigh2=vans
      endif
      write(*,82)
 82   format(1x,'BIM Zeroed manually (=1) or automatically (=0)? ')
      read(*,90)nans
      if (nans.eq.1) then
            write(*,10)
        write(*,100)
        read(*,110)vans
        vhigh3=vans
      endif
 120  return
      end
c
      subroutine scale(iselect)
c
c       To scale any VISAR records given the scaling factor.
c
c       Variables:
c           v1low  - The lowest voltage for trace #1
c           v2low  - The lowest voltage for trace #2
c           blow   - The lowest voltage for the BIM trace
c           vnew   - The voltage for a given data after scaling
c           s      - The manual scaling factor for a given trace
c           iselect - The main menu selection
c
      implicit real*8 (a-h,o-z), integer (i-n)
      common /list1/ signal(10001,3), npsig
      common /list2/ tmpsig(10001,3)
      common /list3/ bimdat(10001,2), npbim
      common /list4/ tmpbim(10001,2)
```

```
c
 5   write(*,10)
 10  format(1x)
     if (iselect.eq.10) then
        ichoice=0
        go to 45
     endif
     write(*,20)
 20  format(//1x,'Choose Option:'
    c     //1x,'  Automatically Scale Traces      = 0'
    c     /1x,'  Manually Scale VISAR Trace #1    = 1'
    c     /1x,'  Manually Scale VISAR Trace #2    = 2'
    c     /1x,'  Manually Scale BIM Trace         = 3'
    c     /1x,'  Done                             = -1'
    c     //)
     write(*,10)
     write(*,30)
 30  format(1x,'Your selection is = ')
     read(*,40)ichoice
 40  format(i2)
     write(*,10)
 45  if (ichoice.eq.0) then
c
c       First, find minimums of each trace
c
        v1low=1.0d0
                v2low=1.0d0
                blow=1.0d0
                do k = 1, npsig
                        if (tmpsig(k,2).lt.v1low)v1low=tmpsig(k,2)
                        if (tmpsig(k,3).lt.v2low)v2low=tmpsig(k,3)
                end do
                do k = 1, npbim
                        if (tmpbim(k,2).lt.blow)blow=tmpbim(k,2)
                end do
c
c     now, scale any two traces to match the third.  Here, the
c     first trace is arbitrarily chosen as the trace to scale to.
c
                do k = 1, npsig
                        signal(k,3)=tmpsig(k,3)*(v1low/v2low)
                end do
                do k = 1, npbim
                        bimdat(k,2)=tmpbim(k,2)*(v1low/blow)
                end do
                go to 300
     endif
     if (ichoice.eq.1) then
                write(*,50)
 50     format(1x,'What is the scaling factor?')
                read(*,60)s
 60     format(f20.16)
                do k = 1, npsig
                        signal(k,2)=tmpsig(k,2)*s
                end do
     endif
     if (ichoice.eq.2) then
                write(*,150)
 150    format(1x,'what is the scaling factor?')
                read(*,160)s
 160    format(f20.16)
                do k = 1, npsig
                        signal(k,3)=tmpsig(k,3)*s
                end do
     endif
     if (ichoice.eq.3) then
                write(*,250)
 250    format(1x,'What is the scaling factor?')
                read(*,260)s
 260    format(f20.16)
                do k = 1, npbim
                        bimdat(k,2)=tmpbim(k,2)*s
                end do
     endif
     if (ichoice.eq.-1) go to 300
     if (iselect.eq.10) go to 300
     go to 5
 300 return
     end
c
     subroutine normal
c
c     To normalize the VISAR records given the same amplitude BIM
c     data.
c
c    Variables:
c
c        volt  - The voltage of the current, scaled data trace
c        bvolt - The BIM voltage at the same time
c        vnew  - The normalized voltage (BIM-corrected)
c
c
     implicit real*8 (a-h,o-z), integer (i-n)
     common /list1/ signal(10001,3), npsig
     common /list2/ tmpsig(10001,3)
     common /list4/ tmpbim(10001,2)

     write(*,10)
 10  format(1x)
     i = 0
     k = 1
 60  i = i + 1
     if (i.ge.npsig) go to 85
 75  if (tmpbim(i,1).eq.tmpsig(i,1)) then
       k = i
       go to 80
     end if
 70  k = k + 1
     if (tmpbim(k,1).eq.tmpsig(i,1)) go to 80
     if (tmpbim(k,1).lt.tmpsig(i,1)) go to 70
     if (tmpbim(k,1).gt.tmpsig(i,1)) then
            i = i + 1
            go to 75
     endif
 80  if (tmpbim(k,2).gt.-1.e-6) go to 60
     signal(i,2)=-1.0d0*(tmpsig(i,2)/tmpbim(i,2))
     if (signal(i,2).ge.10.0) signal(i,2)=9.999
     if (signal(i,2).le.-10.0) signal(i,2)=-9.999
     go to 60
 85  i = 0
     k = 1
 90  i = i + 1
     if (i.ge.npsig) go to 115
 95  if (tmpbim(i,1).eq.tmpsig(i,1)) then
       k = i
       go to 110
     end if
 100 k = k + 1
     if (tmpbim(k,1).eq.tmpsig(i,1)) go to 110
     if (tmpbim(k,1).lt.tmpsig(i,1)) go to 100
     if (tmpbim(k,1).gt.tmpsig(i,1)) then
            i = i + 1
       go to 95
     endif
 110 if (tmpbim(k,2).gt.-1.e-6) go to 90
     signal(i,3)=-1.0d0*(tmpsig(i,3)/tmpbim(i,2))
     if (signal(i,3).ge.10.0) signal(i,3)=9.999
     if (signal(i,3).le.-10.0) signal(i,3)=-9.999
     go to 90
 115 return
     end
c
c
     subroutine contrast(phi)
c
c
c     To calculate the contrast versus time for any two VISAR records
c     given the phase difference between them
c
c    Variables:
c
c        phi   - The phase diff between the signals
c        y1    - The normalized voltage of trace #1
c        y2    - The normalized voltage of trace #2 at the same time
c        ctrast - The contrast at the same time as y1 and y2
c
```

```
c
      implicit real*8 (a-h,o-z), integer (i-n)
      common /list1/ signal(10001,3), npsig
      common /list7/ ctrast(10001)
c
      write(*,10)
 10   format(1x)
      write(*,40)
 40   format(1x,'The phase diff between the signals (deg)? ')
      read(*,45) phi
 45   format(f8.4)
      phi = (4.0d0*datan(1.0d0)*phi)/180.0d0
      write(*,10)
      do i = 1, npsig
       term1=2.0d0*signal(i,2)+1.0d0
       term2=(2.0d0*signal(i,2)+1.0d0)*dcos(phi)
       term3=(2.0d0*signal(i,3)+1.0d0-term2)/dsin(phi)
       ctrast(i)=dsqrt(term1**2+term3**2)
      end do
      return
      end
c
      subroutine velocity(frconst,nwin,phi)
c
c       To calculate the velocity versus time from any two VISAR
c       records given the phase difference between them
c       This program uses 'f' to store the time-history
c       of the fringe count and 'vel' for the velocity history.
c
c
c   Variables:
c
c     refind  - Refractive index of the etalon
c     delta   - Correction factor for etalon dispersion
c     wavel   - Wavelength of the laser light
c     speed   - Speed of light
c     nwin    - Identifies the window material
c     phi     - The phase diff between the signals
c     etal    - The total length of etalon material
c     tau     - The delay time
c     frconst - The fringe constant
c     thet0   - The original phase of the 1st VISAR record
c     veloc   - The particle velocity
c     delfreq - Window correction factor
c     y1      - The normalized voltage for trace #1
c     y2      - The normalized voltage for trace #2
c     con     - The corresponding contrast at the same time
c     n       - Tracks the number of full fringes in the fringe count
c     t       - The time minus tau/2
c     flag    - Tracks whether vel falls outside the window calib. range
c     ff      - The fringe count for the current time (non-array)
c
      implicit real*8 (a-h,o-z), integer (i-n)
      integer flag
      common /list1/ signal(10001,3), npsig
      common /list5/ vel(10001,2)
      common /list6/ f(10001,2)
      common /list7/ ctrast(10001)
c
      pi=4.0d0*datan(1.0d0)
c
c   These constants assume BK-7 etalons
c       refind   - refractive index
c       (1+delta) - etalon dispersion factor
c
      refind=1.5205d0
      delta=0.036
c
c   Assume green light and units of mm/fringe for wavelength
c   and mm/microsec for speed
c
      wavel=0.5145e-3
      speed=2.9979d5
c
      write(*,10)
 10   format(1x)
      write(*,20)
 20   format(1x,'What was the window material?'
c     //1x,'    PMMA                  = 1'
c     /1x,'    Fused Silica          = 2'
c     /1x,'    Z-Cut Sapphire        = 3'
c     /1x,'    Lithium Fluoride      = 4'
c     /1x,'    NONE (Free-Surface Shot)  = 5'
c     //)
      write(*,10)
      write(*,30)
 30   format(1x,'Your selection is = ')
      read(*,35)nwin
 35   format(i2)
      write(*,10)
      write(*,46)
 46   format(1x,'The total length of etalon matl. (inches)? ')
      read(*,48)etal
      etal=etal*25.4d0
 48   format(f6.3)
      write(*,10)
c
      tau=(2.0d0*etal*(refind-(1.0d0/refind)))/speed
      frconst=wavel/(2.0d0*tau)
c
c   Now correct for etalon dispersion
c
      frconst=frconst/(1.0d0+delta)
c
c
      flag = 0
      thet0=0.0
      veloc=0.0d0
      delfreq=0.0d0
c
c    "n" will track the number of full fringes within the
c    fringe count F(t).  Initially, n=0
c
      n=0
c
c    To check for full fringe traversals, we need to know the
c    logical values of the current phase of the signal (thetl)
c    and the logical value of the previous phase (thetlold), and
c    then compare the two.  This is handled by tracking previous
c    thetl's and comparing them with the new one.  As per Barker,
c    anytime the two differ by more than +pi or -pi,
c    a jump is assumed.
c
      do 60 i = 1, npsig-1
c
c   For the starting time, we need the
c   initial phase of record#1.  The initial
c   phase of record#2 is this value plus
c   the given phase difference, phi.  Also,
c   we use this initial phase as a convenient
c   starting point for the tracking of thetl.
c   The logical values of thetl are found
c   by comparing the possible phase phase1
c   with a phase of pi minus phase1
c
100   if (thet0.eq.0.0) then
              phase1=dasin((2.0d0*signal(i,2)+1.0d0)/ctrast(i))
              p1min=pi-phase1
              y2try=0.5*(-1.0d0+ctrast(i)*dsin(phase1+phi))
              y2trymin=0.5*(-1.0d0+ctrast(i)*dsin(p1min+phi))
              diftry=dabs(y2try-signal(i,3))
              diftrymin=dabs(y2trymin-signal(i,3))
              if (diftry.lt.diftrymin) then
                thet0=phase1
              else
                thet0=p1min
              endif
              if (thet0.eq.0.0) thet0=1.e-5
              thetlold=thet0
              go to 60
      endif
c
c  The following uses the second trace's value
c  to determine the proper phase associated with
c  a given level on the first trace.
c
```

```
      y1norm=(2.0d0*signal(i,2)+1.0d0)/ctrast(i)
      y2norm=(2.0d0*signal(i,3)+1.0d0)/ctrast(i)
      thetr=dasin(y1norm)
      pimin=pi-thetr
      y2try=0.5*(-1.0d0+ctrast(i)*dsin(thetr+phi))
      y2trymin=0.5*(-1.0d0+ctrast(i)*dsin(pimin+phi))
      diftry=dabs(y2try-signal(i,3))
      diftrymin=dabs(y2trymin-signal(i,3))
      if (diftry.lt.diftrymin) then
            thetl=thetr
      else
            thetl=pimin
      endif
      diff=dabs(thetl-thetlold)
      if (diff.gt.pi) n=n+1
      thetlold=thetl
      f(i,1)=signal(i,1)-(tau*1.0d-6)/2.0d0
      vel(i,1)=f(i,1)
      f(i,2)=dfloat(n)+(thetl-thet0)/(2.0d0*pi)
      ff = f(i,2)
      call window(veloc,ff,frconst,nwin,delfreq,flag)
      vel(i,2)=(frconst*ff)/(1.0d0+delfreq)
60    continue
      return
      end
c
      subroutine jump(frconst,nwin,npsig)
c
c       This subroutine inserts a jump in the velocity record by
c       including an integer fringe count at the desired location
c       in the data.  It allows the user to view the data before
c       accepting it, in order to zero in on the correct number
c       of lost fringes.
c
c   Variables:
c
c       frconst  - The fringe constant
c       nwin     - Identifies the window material
c       start    - The starting time of the jump
c       end      - The ending time of the jump
c       nfringe  - The number of fringes in the jump
c       f        - The fringe count
c       fstart   - Fringe count at the start of the jump
c       fend     - Fringe count at the end of the jump
c       delfreq  - The window correction factor
c       flag     - Flags if vel falls outside the window calib. range
c       xwidth   - Stores the left-right graphics screen config.
c       yheight  - Stores the up-down graphics screen config.
c       cols     - Stores number of text columns avail. - graphics screen
c       rows     - Stores number of text rows avail. - graphics screen
c       dummy    - Dummy variable for calling certain graphics
routines
c       screen   - Screen configuration used by graphics routines
c       ndselect - Tells plotting subroutines which array to display
c       tstart   - Starting time of data to graphically display
c       tend     - Ending time of data to grahically display
c       sstart   - Lower bound of data to graphically display
c       send     - Upper bound of data to graphically display
c       upleftx  - Upper left x-coord. of box for graphical data display
c       uplefty  -  "    "  y-coord.  "  "  "    "    "    "
c       drightx  - Lower right x-coord. "  "  "    "    "    "
c       drighty  -  "    "   y-coord. "  "  "    "    "    "
c
      implicit double precision (a-h,o-z)
      include 'fgraph.fd'
      integer flag
      integer*2 xwidth, yheight, cols, rows
      integer*2 dummy
      record /videoconfig/ screen
      common screen
      common /list5/ vel(10001,2)
      common /list6/ f(10001,2)
      common /list9/ ftry(10001,2),vtry(10001,2)
c
 5    flag = 0
      veloc=0.0d0
      write(*,10)
 10   format(1x)
```

```
15    write(*,20)
20    format(1x,'What is the starting time of the jump (sec)?')
      read(*,*)start
      write(*,10)
      write(*,30)
30    format(1x,'What is the ending time of the jump (sec)?')
      read(*,*)end
      write(*,10)
      write(*,40)
40    format(1x,'How many fringes to insert? ')
      read(*,*)nfringe
      write(*,10)
      i = 1
      do while (f(i,1) .lt. start)
            ftry(i,1) = f(i,1)
        ftry(i,2) = f(i,2)
        i = i + 1
      end do
      if (i .ge. npsig) then
            write(*,80)
80      format(1x,'Starting time too high for existing data!')
            write(*,10)
        go to 15
      endif
      tstart=f(i,1)
      fstart=f(i,2)
      ftry(i,1) = tstart
      ftry(i,2) = fstart
      j = i
      do while (f(j,1) .lt. end)
        j = j + 1
      end do
      tend=f(j,1)
      fend=f(j,2)+dfloat(nfringe)
      istart=i
      i=i+1
      do while (i.lt.j)
        ftry(i,1)=f(i,1)
        ft=fstart+dfloat(i-istart)*(fend-fstart)/dfloat(j-istart)
        ftry(i,2)=ft
        i=i+1
      end do
      ftry(i,1) = tend
      ftry(i,2) = fend
      i=i+1
      do k = i, npsig
        ftry(k,1) = f(k,1)
        ftry(k,2) = f(k,2)+dfloat(nfringe)
      end do
101   format(f11.9,',',f9.5,',',f11.9,',',f9.5)
      do k = 1, npsig
        vtry(k,1) = ftry(k,1)
        ff = ftry(k,2)
        call window(veloc,ff,frconst,nwin,delfreq,flag)
        vtry(k,2) =(frconst*ff)/(1.0d0+delfreq)
      end do
c
c     Now, to let the user examine the signal and decide whether
c     the number of fringes to insert was correct.
c
175   write(*,10)
      write(*,177)
177   format(1x,'Now, to look at this new velocity record,')
      write(*,10)
      ndselect=5
      call erange(ndselect,tstart,tend,sstart,send)
c
c           First, enter the graphics mode and set the screen to accept
c           its maximum number of colors
c
180   dummy=setvideomode($MAXCOLORMODE)
      call getvideoconfig(screen)
c
c           Now, set up the graphics window
c
      xwidth  = screen.numxpixels
      yheight = screen.numypixels
      cols    = screen.numtextcols
```

```
      rows    = screen.numtextrows
      call setviewport(0,0,xwidth-1,yheight-1)
      call settextwindow(1,1,rows,cols)
c
c               Set the coordinates of the window so that the graph(s)
c               will fit comfortably inside of it.
c
 190  upleftx=tstart-0.5*(tend-tstart)
      uplefty=send+0.1*(send-sstart)
      drightx=tend+0.5*(tend-tstart)
      drighty=sstart-0.9*(send-sstart)
      dummy=setwindow(.TRUE.,upleftx,uplefty,drightx,drighty)
      call plot(ndselect,tstart,tend,sstart,send,rows,cols)
 200  write(*,10)
      write(*,210)
 210  format(1x,'Choose one:'
     c  //1x,'    Select New Region           =1'
     c   /1x,'    Quit Graphics               =2'
     c  //)
      write(*,10)
      write(*,220)
 220  format(1x,'Your Selection Is =')
      read(*,230)nselect
 230  format(i2)
      if (nselect.eq.1) then
         call srange(ndselect,tstart,tend,sstart,send)
      elseif (nselect.eq.2) then
         go to 300
      else
         go to 200
      endif
      go to 180
 300  write(*,10)
      write(*,310)
 310  format(1x,'Choose an option:'
     c    //1x,'    Accept the Change            = 1'
     c     /1x,'    Try Again                    = 2'
     c     /1x,'    Quit (Leave Data As Before)  = 3'
     c    //)
      write(*,10)
      write(*,320)
 320  format(1x,'Your selection is = ')
      read(*,330)iselect
 330  format(i2)
      if (iselect.eq.1) then
        do k = 1, npsig
                    f(k,2) = ftry(k,2)
                    vel(k,2) = vtry(k,2)
             end do
             go to 999
      endif
      if (iselect.eq.2) go to 5
      if (iselect.eq.3) go to 999
      go to 300
 999  return
      stop
      end
c
      subroutine window(veloc,ff,frconst,nwin,delfreq,flag)
c
c     To calculate (delta nu)/(nu0) for the window mat'l
c     used in the experiment.  Since this ratio is dependent
c     on the particle velocity, an iteration must take place
c     This ratio is stored in the variable delfreq.
c
c   Variables:
c
c      tol     - The allowable difference between the test particle
c                velocities
c      nwarn   - Tracks whether the part. vel. is outside calibrated
c                range
c      velcurr - The previous velocity
c      veltest - A temporary, iterated velocity
c      flag    - Tells whether a velocity falls outside the exptl.
c                data range used to find the window correction factors
c
c
      implicit double precision (a-h,o-z)
      integer flag
      tol=0.001
      flag=0
      nwarn=0
      velcurr=veloc
      if (nwin.eq.1) go to 10
      if (nwin.eq.2) go to 50
      if (nwin.eq.3) go to 100
      if (nwin.eq.4) go to 150
      if (nwin.eq.5) go to 7
 7    delfreq=0.0d0
      go to 500
c
c    Now, for the PMMA window
c
 10   delfreq=(-0.0658*velcurr)+(0.0762*velcurr**2)+
     c (0.0731*velcurr**3)
      veltest=(ff*frconst)/(1.0d0+delfreq)
      diff=dabs(veltest-velcurr)
      if (diff.lt.tol) go to 30
      velcurr=0.5d0*(velcurr+veltest)
      go to 10
 30   if (veltest.gt.0.6.and.flag.eq.0) nwarn=1
      go to 500
c
c    now, for the fused silica window
c
 50   if (velcurr.lt.0.06) then
               delfreq=0.0d0
      else
               delfreq=0.0315+(0.002316/velcurr)
      endif
      veltest=(ff*frconst)/(1.0d0+delfreq)
      diff=dabs(veltest-velcurr)
      if (diff.lt.tol) go to 70
      velcurr=0.5d0*(velcurr+veltest)
      go to 50
 70   if (veltest.gt.0.6.and.flag.eq.0) nwarn=1
      go to 500
c
c    Now, for the z-cut sapphire window
c
 100  delfreq=0.7901-(0.1983*velcurr)+(0.2013*velcurr**2)
      veltest=(ff*frconst)/(1.0d0+delfreq)
      diff=dabs(veltest-velcurr)
      if (diff.lt.tol) go to 120
      velcurr=0.5d0*(velcurr+veltest)
      go to 100
 120  if (veltest.gt.0.32.and.flag.eq.0) nwarn=1
      go to 500
c
c    Now, for the lithium fluoride window
c
 150  delfreq=0.2566+(0.0226*velcurr)
      veltest=(ff*frconst)/(1.0d0+delfreq)
      diff=dabs(veltest-velcurr)
      if (diff.lt.tol) go to 170
      velcurr=0.5d0*(velcurr+veltest)
      go to 150
 170  if (veltest.gt.1.5.and.flag.eq.0) nwarn=1
 500  if (nwarn.eq.1) then
               write(*,510)
               flag = 1
 510       format(1x,'Warning: Particle Velocity Outside Calibration
     c Range of the Window!!')
      endif
      return
      end
c
      subroutine storeold(npsig, npbim)
c
c     subroutine to bring tmp-->old (i.e. storing the tmp data).
c
      implicit double precision (a-h,o-z)
c
      common /list2/ tmpsig(10001,3)
      common /list4/ tmpbim(10001,2)
      common /list8/ oldsig(10001,3), oldbim(10001,2)
```

```
c
      do k = 1,npsig
            oldsig(k,2) = tmpsig(k,2)
            oldsig(k,3) = tmpsig(k,3)
      end do
      do k = 1,npbim
            oldbim(k,1) = tmpbim(k,1)
            oldbim(k,2) = tmpbim(k,2)
      end do
      return
      end
c
      subroutine undo
c
c     subroutine to bring old-->tmp and tmp-->signal
c
      implicit double precision (a-h,o-z)
      common /list1/ signal(10001,3), npsig
      common /list2/ tmpsig(10001,3)
      common /list3/ bimdat(10001,2), npbim
      common /list4/ tmpbim(10001,2)
      common /list8/ oldsig(10001,3), oldbim(10001,2)
c
      do i = 1, npsig
            signal(i,2) = tmpsig(i,2)
            signal(i,3) = tmpsig(i,3)
            tmpsig(i,2) = oldsig(i,2)
            tmpsig(i,3) = oldsig(i,3)
      end do
      do i = 1, npbim
            bimdat(i,2) = tmpbim(i,2)
            tmpbim(i,2) = oldbim(i,2)
      end do
      return
      end
c
      subroutine reset(vis1,vis2,bim)
c
c     subroutine to bring original data to signal and tmp
c
      implicit double precision (a-h,o-z)
      character * 40 vis1, vis2, bim
      common /list1/ signal(10001,3), npsig
      common /list2/ tmpsig(10001,3)
      common /list3/ bimdat(10001,2), npbim
      common /list4/ tmpbim(10001,2)
      common /list8/ oldsig(10001,3), oldbim(10001,2)
c
      open(unit=1,file=vis1,status='old')
      open(unit=2,file=vis2,status='old')
      open(unit=3,file=bim,status='old')
c
      i = 1
      read(1,*,end=3000) a, b
      read(2,*,end=3000) c, d
5     tmpsig(i,1) = a
            tmpsig(i,2) = b
            tmpsig(i,3) = d
            signal(i,1) = a
            signal(i,2) = b
            signal(i,3) = d
            oldsig(i,1) = a
            i = i + 1
            read(1,*,end=3000) a, b
            read(2,*,end=3000) c, d
      goto 5
3000  npsig = i-1
      i = 1
      read(3,*,end=3100) a, b
30    bimdat(i,1) = a
            bimdat(i,2) = b
            tmpbim(i,1) = a
            tmpbim(i,2) = b
            oldbim(i,2) = b
            i = i + 1
            read(3,*,end=3100) a, b
      goto 30
3100  npbim = i-1
      close(1)
      close(2)
      close(3)
      return
      end
c
      subroutine writedata
c
c     subroutine to write the current data to disk
c
      implicit double precision (a-h,o-z)
      common /list1/ signal(10001,3), npsig
      common /list3/ bimdat(10001,2), npbim
c
      open(unit=14,file='signals.dat',status='unknown')
      open(unit=15,file='bim.dat',status='unknown')
c
      write(14,10) ((signal(i,j),j=1,3),i=1,npsig)
      close(14)
      write(15,20) ((bimdat(i,j),j=1,2),i=1,npbim)
      close(15)
10    format(2(f11.9,','),f9.5)
20    format(f11.9,',',f9.5)
      return
      end
c
      subroutine writevel(npsig)
c
c     subroutine to write velocity data to disk
c
      implicit double precision (a-h,o-z)
      integer npsig
      common /list5/ vel(10001,2)
      open(unit=19,file='velocity.dat',status='unknown')
c
      do i = 2,npsig-1
       write(19,10) (vel(i,j),j=1,2)
      end do
      close(19)
10    format(f11.9,',',f9.5)
      return
      end
c
      subroutine writecontrast(npsig)
c
c     subroutine to write contrast to disk
c
      implicit double precision (a-h,o-z)
      integer npsig
      common /list7/ ctrast(10001)
c
      open(unit=19,file='contrast.dat',status='unknown')
      write(19,10) (ctrast(i),i=2,npsig-1)
      close(19)
10    format(f9.5)
      return
      end
c
      subroutine writefringe(npsig)
c
c     subroutine to write fringe count data to disk
c
      implicit double precision (a-h,o-z)
      integer npsig
      common /list6/ f(10001,2)
c
      open(unit=19,file='fringec.dat',status='unknown')
      write(19,10) ((f(i,j),j=1,2),i=2,npsig-1)
      close(19)
10    format(f11.9,',',f9.5)
      return
      end
c
      subroutine graphics
c
c     This is the main menu graphics subroutine
c
c     Variables:
```

```
c       xwidth  - Stores the left-right graphics screen config.
c       yheight - Stores the up-down graphics screen config.
c       cols    - Stores number of text columns avail. - graphics screen
c       rows    - Stores number of text rows avail. - graphics screen
c       dummy   - Dummy variable for calling certain graphics
routines
c       screen  - Screen configuration used by graphics routines
c       wxy     - Graphics cursor xy pos. prior to 'moveto' command
c       curpos  - Text cursor xy pos. prior to 'settextposition' comm.
c       ndselect - Tells plotting subroutines which array to display
c       tstart  - Starting time of data to graphically display
c       tend    - Ending time of data to grahically display
c       sstart  - Lower bound of data to graphically display
c       send    - Upper bound of data to graphically display
c       upleftx - Upper left x-coord. of box for graphical data display
c       uplefty -  "    "  y-coord.  "  "  "    "    "    "
c       drightx - Lower right x-coord. "  "  "    "    "    "
c       drighty -  "    "   y-coord. "  "  "    "    "    "
c
      implicit double precision (a-h,o-z)
      include 'fgraph.fd'
      integer*2 xwidth, yheight, cols, rows
      integer*2 dummy
      record /videoconfig/ screen
      common screen
      common /list1/ signal(10001,3),npsig
      common /list3/ bimdat(10001,2),npbim
      common /list5/ vel(10001,2)
      common /list7/ ctrast(10001)
c
      nselect=0
 5    write(*,10)
 10   format(1x)
      write(*,20)
 20   format(1x,'Pick data to plot:'
     c           //1x,'    BIM and Both Signals    =1'
     c            /1x,'    Both Signals Only      =2'
     c            /1x,'    Contrast               =3'
     c     /1x,'    Velocity               =4'
     c     /1x,'    Quit                   =0'
     c            //)
      write(*,10)
      write(*,30)
 30   format(1x,'Your Selection Is =')
      read(*,40)ndselect
 40   format(i2)
      if (ndselect.eq.0) go to 100
 45   write(*,10)
      write(*,50)
 50   format(1x,'Plot Entire History (=0) or Selected Region (=1)? ')
      read(*,40)ntselect
      if (ntselect.eq.0) then
              call erange(ndselect,tstart,tend,sstart,send)
      elseif (ntselect.eq.1) then
         call srange(ndselect,tstart,tend,sstart,send)
      else
              go to 45
      endif
c
c             First, enter the graphics mode and set the screen to accept
c             its maximum number of colors
c
 55   dummy=setvideomode($MAXCOLORMODE)
      call getvideoconfig(screen)
c
c             Now, set up the graphics window
c
      xwidth  = screen.numxpixels
      yheight = screen.numypixels
      cols    = screen.numtextcols
      rows    = screen.numtextrows
      call setviewport(0,0,xwidth-1,yheight-1)
      call settextwindow(1,1,rows,cols)
c
c             Set the coordinates of the window so that the graph(s)
c             will fit comfortably inside of it.
c
 60   upleftx=tstart-0.5*(tend-tstart)
      uplefty=send+0.1*(send-sstart)
      drightx=tend+0.5*(tend-tstart)
      drighty=sstart-0.9*(send-sstart)
      dummy=setwindow(.TRUE.,upleftx,uplefty,drightx,drighty)
c
c     Now, plot everything inside the graphics window and add labels.
c
      call plot(ndselect,tstart,tend,sstart,send,rows,cols)
 65   write(*,10)
      write(*,70)
 70   format(1x,'Choose one:'
     c     //1x,'    Select New Region           =1'
     c      /1x,'    Back to Main Graphics Menu   =2'
     c      /1x,'    Quit Graphics               =0'
     c     //)
      write(*,10)
      write(*,80)
 80   format(1x,'Your Selection Is =')
      read(*,40)nselect
      if (nselect.eq.1) then
         call srange(ndselect,tstart,tend,sstart,send)
      elseif (nselect.eq.2) then
         go to 5
      elseif (nselect.eq.0) then
         go to 100
      else
         go to 65
      endif
      go to 55
 100  return
      end
c
      subroutine plot(ndselect,tstart,tend,sstart,send,rows,cols)
c
c     Plots data into the graphics window and adds labels
c
c     Variables:
c        str   - Text string variable used by 'outtext' to output text
c        nr    - Row number location desired for text output
c        nc    - Col. number location desired for text output
c
      implicit double precision (a-h,o-z)
      include 'fgraph.fd'
      integer*2 dummy,i,rows,cols
      character*10 str
      record /videoconfig/ screen
      record /wxycoord/    wxy
      record /rccoord/ curpos
      common screen
      common /list1/ signal(10001,3), npsig
      common /list3/ bimdat(10001,2), npbim
      common /list5/ vel(10001,2)
      common /list7/ ctrast(10001)
      common /list9/ ftry(10001,2),vtry(10001,2)
c
c     Draw a bordered rectangle around the graph.
c
      dummy = setcolor(15)
      dummy = rectangle_w($GBORDER,tstart,send,tend,sstart)
c
c             Plot the points
c
      if (ndselect.eq.3) go to 100
      if (ndselect.eq.4) go to 200
      if (ndselect.eq.5) go to 250
c
c             Plotting both signals and, in the case of selection 1, the
BIM
c
      dummy=settextcolor(15)
      nr=5
      nc=62
      call settextposition(nr,nc,curpos)
      str='Signal #1'
      call outtext(str)
      do 10 i=1,npsig
              if (signal(i,1).gt.tstart) go to 20
 10   continue
```

```
20  istart=i
    call moveto_w(signal(istart,1),signal(istart,2),wxy)
    do 30 i=istart,npsig
              if (signal(i,1).gt.tend) go to 40
              dummy=lineto_w(signal(i,1),signal(i,2))
30  continue
40  iend=i-1
    dummy=setcolor(10)
    dummy=settextcolor(10)
    nr=6
    nc=62
    call settextposition(nr,nc,curpos)
    str='Signal #2'
    call outtext(str)
    call moveto_w(signal(istart,1),signal(istart,3),wxy)
    istart=istart+1
    do 50 i=istart,iend
              dummy=lineto_w(signal(i,1),signal(i,3))
50  continue
    if (ndselect.eq.2) go to 400
    dummy=setcolor(14)
    dummy=settextcolor(14)
    nr=7
    nc=62
    call settextposition(nr,nc,curpos)
    str='BIM'
    call outtext(str)
    do 60 i=1,npbim
              if (bimdat(i,1).gt.tstart) go to 70
60  continue
70  call moveto_w(bimdat(i,1),bimdat(i,2),wxy)
    istart=i+1
    do 80 i=istart,npbim
              if (bimdat(i,1).gt.tend) go to 90
              dummy=lineto_w(bimdat(i,1),bimdat(i,2))
80  continue
90  go to 400
c
c             Now, plot the contrast function
c
100 dummy=settextcolor(15)
    nr=5
    nc=62
    call settextposition(nr,nc,curpos)
    str='Contrast'
    call outtext(str)
    do 110 i=1,npsig
              if (signal(i,1).gt.tstart) go to 120
110 continue
120 istart=i
    call moveto_w(signal(istart,1),ctrast(istart),wxy)
    do 130 i=1,npsig-1
       if (signal(i+1,1).gt.tend) go to 400
              dummy=lineto_w(signal(i,1),ctrast(i))
130 continue
140 go to 400
c
c             Now, plot the velocity history
c
200 dummy=settextcolor(15)
    nr=5
    nc=62
    call settextposition(nr,nc,curpos)
    str='Velocity'
    call outtext(str)
    do 210 i=1,10001
              if (vel(i,1).gt.tstart) go to 220
210 continue
220 istart=i+1
    call moveto_w(vel(istart,1),vel(istart,2),wxy)
    istart=istart+1
    do 230 i=istart,10001
       if (vel(i,1).gt.tend) go to 400
       if (vel(i,1).lt.vel(i-1,1)) go to 400
              dummy=lineto_w(vel(i,1),vel(i,2))
230 continue
    go to 400
c
```

```
c     Now, plot the fringe(s)-added velocity history
c
250 dummy=settextcolor(15)
    nr=5
    nc=62
    call settextposition(nr,nc,curpos)
    str='Velocity'
    call outtext(str)
    do 260 i=1,10001
       if (vtry(i,1).gt.tstart) go to 270
260 continue
270 istart=i+1
    call moveto_w(vtry(istart,1),vtry(istart,2),wxy)
    istart=istart+1
    do 280 i=istart,10001
       if (vtry(i,1).gt.tend) go to 400
       if (vtry(i,1).lt.vtry(i-1,1)) go to 400
       dummy=lineto_w(vtry(i,1),vtry(i,2))
280 continue
400 continue
c
c    Print the limits and axis labels on the screen
c
    dummy=settextcolor(15)
    nr=int(rows/4.)-5
    nc=int(cols/8.)
    call settextposition(nr,nc,curpos)
    write(str,'(e10.3)')send
    call outtext(str)
    nr=int(3.*rows/4.)-5
    call settextposition(nr,nc,curpos)
    write(str,'(e10.3)')sstart
    call outtext(str)
    nr=int(7.*rows/8.)-8
    nc=int(cols/4.)-5
    call settextposition(nr,nc,curpos)
    write(str,'(e10.3)')tstart
    call outtext(str)
    nc=int(3.*cols/4.)-5
    call settextposition(nr,nc,curpos)
    write(str,'(e10.3)')tend
    call outtext(str)
    nr=9
    nc=10
    call settextposition(nr,nc,curpos)
    str='Level'
    call outtext(str)
    nr=19
    nc=39
    call settextposition(nr,nc,curpos)
    str='Time'
    call outtext(str)
c
c             Pause here and wait for user to view graph
c
    write(*,410)
410 format(1x)
    write(*,420)
420 format(1x,'Activate the graphics window to view the graph...')
    write(*,410)
    write(*,430)
430 format(1x,'Press RETURN to continue...')
    read(*,*)
    dummy=setvideomode($DEFAULTMODE)
    return
    end
c
    subroutine srange(ndselect,tstart,tend,sstart,send)
c
c     This subroutine finds the ranges for plotting
c             user-selected parts of a trace
c
c     Variables:
c       tstart  - Starting time of data to graphically display
c       tend    - Ending time of data to grahically display
c       sstart  - Lower bound of data to graphically display
c       send    - Upper bound of data to graphically display
c       slow    - Lowest signal level in user-selected time range
```

```
c        shigh    - Highest "     "    "      "         "    "
c
      implicit double precision (a-h,o-z)
      common /list1/ signal(10001,3),npsig
      common /list3/ bimdat(10001,2),npbim
      common /list5/ vel(10001,2)
      common /list7/ ctrast(10001)
      common /list9/ ftry(10001,2),vtry(10001,2)
c
      write(*,10)
 10   format(1x)
      write(*,20)
 20   format(1x,'Input Starting Time =:')
      read(*,*)tstart
      write(*,10)
      write(*,30)
 30   format(1x,'Input Ending Time =:')
      read(*,*)tend
      if (ndselect.eq.2) go to 100
      if (ndselect.eq.3) go to 200
      if (ndselect.eq.4) go to 300
      if (ndselect.eq.5) go to 350
c
c     Here, find the range for both signals and the BIM....
c
      j=0
      do 40 i=1,npsig
         if (signal(i,1).gt.tend) go to 35
         if (signal(i,1).lt.tstart) go to 35
         if (j.eq.0) then
            j=1
            slow=signal(i,2)
            shigh=signal(i,2)
         endif
         if(slow.gt.signal(i,2)) slow=signal(i,2)
         if(slow.gt.signal(i,3)) slow=signal(i,3)
         if(shigh.lt.signal(i,2)) shigh=signal(i,2)
         if(shigh.lt.signal(i,3)) shigh=signal(i,3)
 35      if (i.gt.npbim) go to 40
         if (bimdat(i,1).gt.tend) go to 40
         if (bimdat(i,1).lt.tstart) go to 40
         if (j.eq.0) then
            j=1
            slow=bimdat(i,2)
            shigh=bimdat(i,2)
         endif
         if (slow.gt.bimdat(i,2)) slow=bimdat(i,2)
         if (shigh.lt.bimdat(i,2)) shigh=bimdat(i,2)
 40   continue
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for both signals....
c
 100  j=0
      do 140 i=1,npsig
         if (signal(i,1).gt.tend) go to 140
         if (signal(i,1).lt.tstart) go to 140
         if (j.eq.0) then
            j=1
            slow=signal(i,2)
            shigh=signal(i,2)
         endif
         if(slow.gt.signal(i,2)) slow=signal(i,2)
         if(slow.gt.signal(i,3)) slow=signal(i,3)
         if(shigh.lt.signal(i,2)) shigh=signal(i,2)
         if(shigh.lt.signal(i,3)) shigh=signal(i,3)
 140  continue
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the contrast function...
c
 200  j=0
      do 240 i=1,npsig
         if (signal(i,1).gt.tend) go to 240
         if (signal(i,1).lt.tstart) go to 240
         if (j.eq.0) then
            j=1
            slow=ctrast(i)
            shigh=ctrast(i)
         endif
         if(slow.gt.ctrast(i)) slow=ctrast(i)
         if(shigh.lt.ctrast(i)) shigh=ctrast(i)
 240  continue
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the velocity function...
c
 300  j=0
      do 340 i=1,npsig
         if (vel(i,1).gt.tend) go to 340
         if (vel(i,1).lt.tstart) go to 340
         if (j.eq.0) then
            j=1
            slow=vel(i,2)
            shigh=vel(i,2)
         endif
         if(slow.gt.vel(i,2)) slow=vel(i,2)
         if(shigh.lt.vel(i,2)) shigh=vel(i,2)
 340  continue
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the fringe(s)-inserted velocity
c     function...
c
 350  j=0
      do 390 i=1,npsig
         if (vtry(i,1).gt.tend) go to 390
         if (vtry(i,1).lt.tstart) go to 390
         if (j.eq.0) then
            j=1
            slow=vtry(i,2)
            shigh=vtry(i,2)
         endif
         if(slow.gt.vtry(i,2)) slow=vtry(i,2)
         if(shigh.lt.vtry(i,2)) shigh=vtry(i,2)
 390  continue
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
 400  return
      end
c
      subroutine erange(ndselect,tstart,tend,sstart,send)
c
c      This subroutine finds the ranges for the entire
c      data file
c
c      Variables:
c         tstart   - Starting time of data to graphically display
c         tend     - Ending time of data to grahically display
c         sstart   - Lower bound of data to graphically display
c         send     - Upper bound of data to graphically display
c         tlow     - Earliest time with data
c         thigh    - Latest time with data
c         slow     - Lowest signal level
c         shigh    - Highest "      "
c
      implicit double precision (a-h,o-z)
      common /list1/ signal(10001,3),npsig
      common /list3/ bimdat(10001,2),npbim
      common /list5/ vel(10001,2)
      common /list7/ ctrast(10001)
      common /list9/ ftry(10001,2),vtry(10001,2)
c
      write(*,10)
 10   format(1x)
      if (ndselect.eq.2) go to 100
      if (ndselect.eq.3) go to 200
      if (ndselect.eq.4) go to 300
```

```
      if (ndselect.eq.5) go to 370
c
c     Here, find the range for both signals and the BIM....
c
      tlow=signal(1,1)
      if (tlow.lt.bimdat(1,1)) tlow=bimdat(1,1)
      thigh=tlow
      slow=signal(1,2)
      if (slow.gt.signal(1,3)) slow=signal(1,3)
      if (slow.gt.bimdat(1,2)) slow=bimdat(1,2)
      shigh=signal(1,2)
      if (shigh.lt.signal(1,3)) shigh=signal(1,3)
      if (shigh.lt.bimdat(1,2)) shigh=bimdat(1,2)
      do 60 i=2,npsig
         if (signal(i,1).gt.thigh) thigh=signal(i,1)
         if (signal(i,2).lt.slow) slow=signal(i,2)
         if (signal(i,2).gt.shigh) shigh=signal(i,2)
         if (signal(i,3).lt.slow) slow=signal(i,3)
         if (signal(i,3).gt.shigh) shigh=signal(i,3)
         if (i.gt.npbim) go to 60
         if (bimdat(i,2).lt.slow) slow=bimdat(i,2)
         if (bimdat(i,2).gt.shigh) shigh=bimdat(i,2)
 60   continue
      tstart=tlow-0.1*(thigh-tlow)
      tend=thigh+0.1*(thigh-tlow)
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for both signals....
c
 100  tlow=signal(1,1)
      thigh=tlow
      slow=signal(1,2)
      if (slow.gt.signal(1,3)) slow=signal(1,3)
      shigh=signal(1,2)
      if (shigh.lt.signal(1,3)) shigh=signal(1,3)
      do 160 i=2,npsig
         if (signal(i,1).gt.thigh) thigh=signal(i,1)
         if (signal(i,2).lt.slow) slow=signal(i,2)
         if (signal(i,2).gt.shigh) shigh=signal(i,2)
         if (signal(i,3).lt.slow) slow=signal(i,3)
         if (signal(i,3).gt.shigh) shigh=signal(i,3)
 160  continue
      tstart=tlow-0.1*(thigh-tlow)
      tend=thigh+0.1*(thigh-tlow)
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the contrast function....
c
 200  tlow=signal(1,1)
      thigh=tlow
      slow=ctrast(1)
      shigh=slow
      do 260 i=2,npsig
         if (signal(i,1).gt.thigh) thigh=signal(i,1)
         if (ctrast(i).lt.slow) slow=ctrast(i)
         if (ctrast(i).gt.shigh) shigh=ctrast(i)
 260  continue
      tstart=tlow-0.1*(thigh-tlow)
      tend=thigh+0.1*(thigh-tlow)
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the velocity function....
c
 300  tlow=vel(2,1)
      thigh=tlow
      slow=vel(2,2)
      shigh=slow
      do 360 i=3,npsig
         if (vel(i,1).gt.thigh) thigh=vel(i,1)
         if (vel(i,2).lt.slow) slow=vel(i,2)
         if (vel(i,2).gt.shigh) shigh=vel(i,2)
 360  continue
      tstart=tlow-0.1*(thigh-tlow)
      tend=thigh+0.1*(thigh-tlow)
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
      go to 400
c
c     Here, find the range for the fringe(s)-inserted velocity
c     function....
c
 370  tlow=vtry(2,1)
      thigh=tlow
      slow=vtry(2,2)
      shigh=slow
      do 380 i=3,npsig
         if (vtry(i,1).gt.thigh) thigh=vtry(i,1)
         if (vtry(i,2).lt.slow) slow=vtry(i,2)
         if (vtry(i,2).gt.shigh) shigh=vtry(i,2)
 380  continue
      tstart=tlow-0.1*(thigh-tlow)
      tend=thigh+0.1*(thigh-tlow)
      sstart=slow-0.1*(shigh-slow)
      send=shigh+0.1*(shigh-slow)
 400  return
      end
```