Internal report 95--03

# Operation of VISAR System

Gang Yuan

May 1995

Shock Dynamics Center
Washington State University
Pullman, WA 99164-2814

## I. Introduction

VISAR is a worldwide used velocity measurement system. The advantages of the VISAR are the high temporal resolution, reliable and able to measure whole history (no matter the loading or unloading path). However, explicit drawbacks are in the VISAR system. Unlike the gauge measurements, the data obtained from VISAR measurements are not straightforward. It is not easy to get the fringes in the experiment (precise optical alignment is necessary). The more difficult thing is to get correct velocity data from the measured fringes. These drawbacks may limit people to use it. The VISAR system has been set up at the Shock Dynamics Center. The basic theory and data reduction have been discussed in Ref. [1]. This report will be a supplement material to Ref. [1]. The purpose of this report is to give user a working knowledge and basic idea to operate VISAR system. Emphasized will be the data reduction part and the parts which were not fully discussed in Ref.[1]. For the convenience, we use all symbols similar to Ref.[1].

A brief introduction of the theory of VISAR will be given in Sec. II. We start from the Doppler theory and let user quickly understand the basic theory of the VISAR. The construction of VISAR is described in Sec. III. We will discuss the resolution of VISAR system, which is critical for the experiment, in Sec. IV. In Sec V, we will emphasize the data reduction to guide user transferring experimental data (fringes) to particle velocity, and predicting fringes from the assumed particle velocity.

## II. Theory of the VISAR

The basic idea of VISAR was derived from the Doppler theory. When an incident light (with the wavelength $\lambda$) is reflected by a moving body with a velocity u, as shown in Fig.1, the reflected light will have a wavelength shift ($\lambda$ to $\lambda+\Delta\lambda$).

This is so called Doppler shift. The quantitative relations between the velocity u and the wavelength shift $\Delta\lambda$ can be expressed by equation (1).

$$\Delta\lambda/\lambda = -2u/c \qquad (1)$$

where c is light speed. Ideally if one measures the wavelength shift $\Delta\lambda$, one is able to know the particle velocity of the moving body. The earliest design of the Sandia VISAR system[2] is shown in Fig. 2. Two beams enter the photomultiplier with traveling different optical paths. One beam has a delay

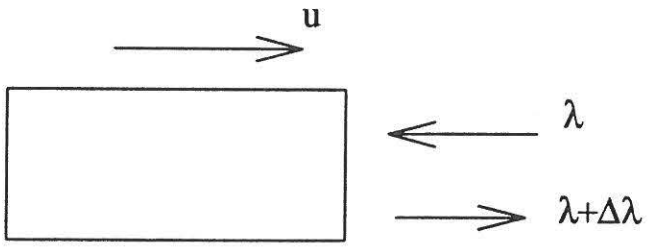



Figure 1

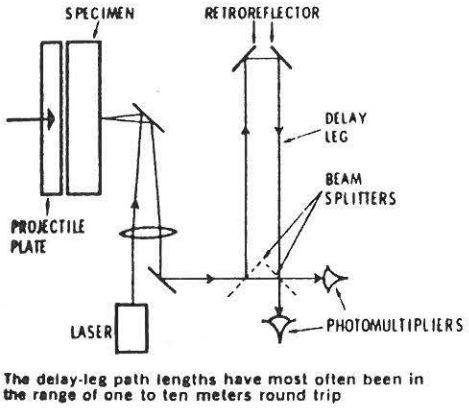



Figure 2

time ($\tau$). These two beams carry the different information of the target surface. One is on the time of t- $\tau$ and the other one is at the time of t. The two beams interfere at the place of potomultiplier resulting the fringes. If the target surface moves during the delay time $\tau$, this movement will result the fringes at the recording plane. Next step is to establish the relation between the fringes and particle velocity. We start from Doppler theory:

$$\Delta\lambda = -(2\lambda/c)/u \qquad (2)$$

The optical length of the delay leg is:

$$N\lambda=c\tau \qquad (3)$$

where N is the number of the fringes, $\tau$ is the delay time. Equation (3) means that the optical length of the delay leg is N times wavelength. Differentiating equation (3), we have:

$$\Delta N=-(c\,\tau/\lambda^2)\Delta\lambda(t) \qquad (4)$$

Substitute (2) in to (4), we get:

$$u=\lambda\Delta N/2\tau \qquad (5)$$

Here $\Delta N$ is the change of the fringes during time $\tau$. We can also call it fringe frequency. Let $\Delta N=F(t)$, equation (5) becomes:

$$u=\lambda F(t)/2\tau \qquad (6)$$

Since u is the average value of the velocity in the time interval $(t-\tau, t)$, it can be approximated by $u(t-\tau/2)$:

$$u(t-\tau/2)=\lambda F(t)/2\tau \qquad (7)$$

Equation (7) is the basic relation between the particle velocity and the fringe frequency. If window materials are used, an additional frequency shift $\Delta\nu$ is created. Furthermore, ~~when~~ due to Doppler shift light entering the etalon in VISAR, a very slight change ($\delta$) will happen due to the refraction index of elaton. These two factors can be modified as (detailed explanations can be found in [1]):

$$u(t-\tau/2)=\lambda F(t)/(2\tau(1+\delta)(1+\Delta\nu/\nu_0)) \qquad (8)$$

Equation (8) is a key relation in VISAR data analysis.

## III. Construction of the VISAR

The basic construction of VISAR system has been described in [1]. Some modifications have been made in the current configuration. The

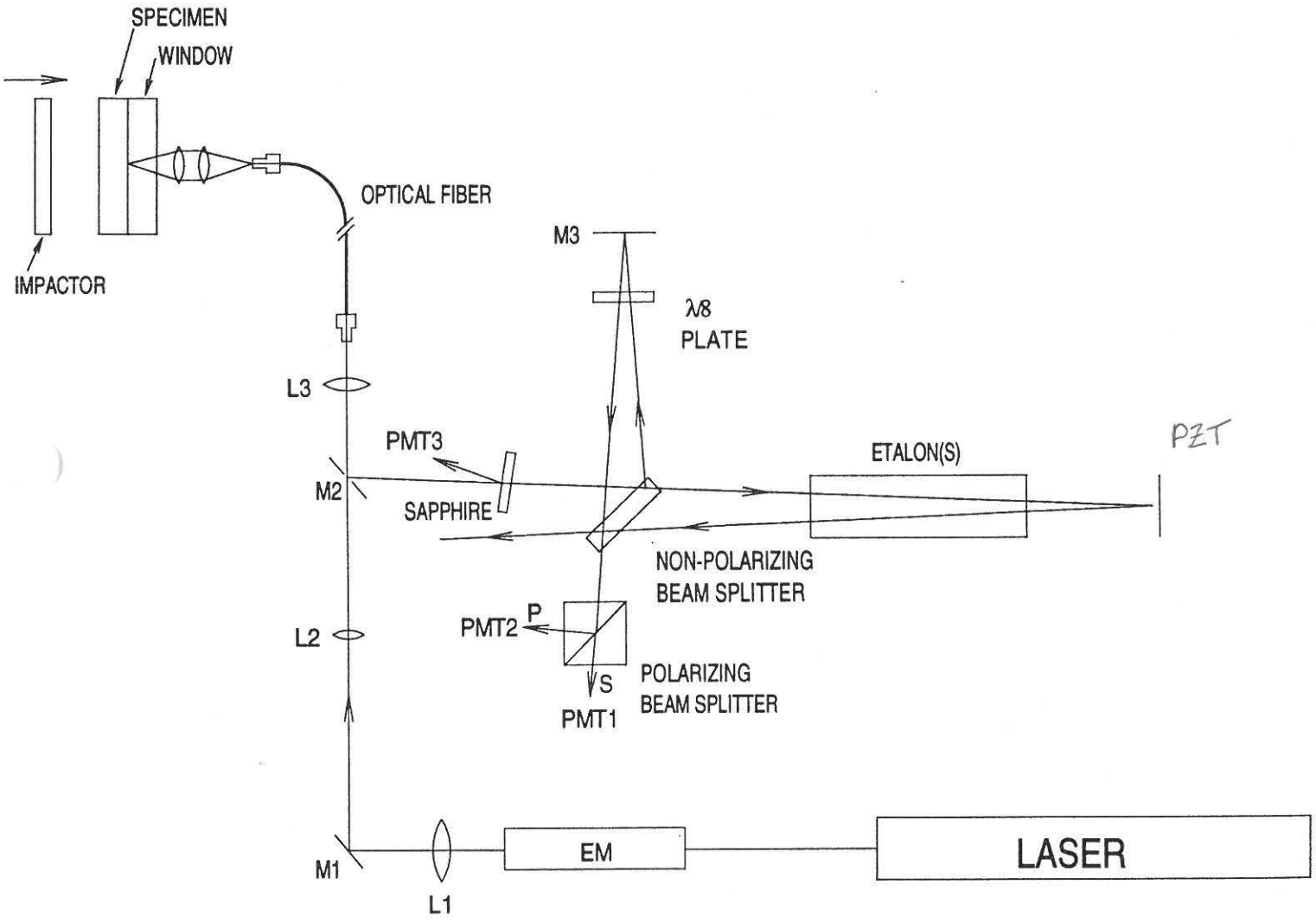

SPECIMEN
WINDOW
IMPACTOR
OPTICAL FIBER
M3
λ/8
PLATE
L3
PMT3
PZT
ETALON(S)
M2
SAPPHIRE
NON-POLARIZING
BEAM SPLITTER
L2
PMT2
P
S
POLARIZING
BEAM SPLITTER
PMT1
M1
L1
EM
LASER

Figure 3

modifications are explained as the following. The non-polarizing beam splitter (prism) has been replaced by a plane beam splitter. The distances from nonploarzing beam splitter to the PZT and the M3 are 14.50 and 10.25 inches, respectively. Electronic modulator was used between laser and M1 as shown in Fig. 3. PMT1 and PMT2 have been changed from the old type 62-SPL1044 to the new type VALYN. PMT3 is still the old type of 62-SPL1044. 12 inches etalon was used. Schematic of the VISAR was expressed in Figure 2.

IV. Resolution of the current VISAR system

The velocity resolutions of VISAR is determined by the delay time $\tau$ in the delay leg. The optical path in the delay leg is longer than that in the non-delay leg since light travels slower inside the etalon and the different distances of the two legs. Now we decide the delay times in the two legs. (note: the equation (2.21) in [1] should be used with the conditions which geometrical distance in the delay leg h would equal nh', where h' is the geometrical distance in the no delay leg. Strictly speaking, use of (2.21) in [1] is incorrect). We calculated the time of the beams in the different legs, respectively.

In the delay leg, the time of traveling light is:

$$t = 2hn/c + 2\Delta X/c \qquad (9)$$

where the first part is the traveling time in the etalon and the second part is the light traveling time in the rest part of the delay leg. In the present configuration, $\Delta x$ is 2.5 inches

In the no-delay leg, the time of traveling light is:

$$t' = 2h'/c \qquad (10)$$

and delay time $\tau$ between the two legs is

$$\tau = t - t' \qquad (11)$$

If $\Delta x=0$ and $h=nh'$, $\tau$ can be expressed by (2.21) in [1]. In the current VISAR system, h=12 inches, h'=10.25 inches, $\Delta x$=14.5'-12'=2.5 inches. For the BK-7 etalon, n is 1.5205. From equations (9-11) we have t=3.51 ns, t'=1.74 ns and $\tau$ = 1.77 ns. Consider the Ar laser ($\lambda$=514.5 nm) and neglect

1.78 ns

the small modifications of the dispersion of window material and etalon, we get u(t)=0.145 mm/μs from equation (7). This means that change of one fringe will correspond to velocity change of 0.145 mm/μs. Barker established that the value F(t) can generally be determined to at least 0.1 fringe [2]. This means that the velocity resolution in the current VISAR system is about 0.0145 mm/μs. Actually, the resolution depends on the particular system, such as response of PMT and the resolution of digitized scope (DSA). In a SiC experiment (Figure 5), the resolution of the system is limited by these factors at the shock front where 30 data points were recorded in one fringe. So we conclude that resolution in this particular experiment is 3% fringe.

V. Data reduction

Data reduction is a crucial part of operating the VISAR system. From equation (8), one can not figure out the increase or decrease of the particle velocities from the change of the fringes. In order to distinguish that, a polarized beam splitter was used in the VISAR as shown in Fig. 3. This beam splitter separate the beam to S and P components, which have a $90^{o}$ phase different between each other. If S and P beams are recorded by representing DATA1 and DATA2, DATA1 leading DATA2 corresponds the increase of particle velocity. While DATA2 leading DATA1 corresponds the decrease of particle velocity.

Data reduction procedures

1. Record of signals

In the experiments, three signals are recorded, DATA1, DATA2 and beam intensity BIM by PMT1, PMT2 and PMT3, respectively.

2. Normalization of the two fringe signals.

Assuming the recorded fringe (DATA1 and DATA2) amplitude relationship have been obtained. We have to eliminate the effects of beam intensity (BIM) variation by dividing the recorded DATA1 and DATA2 trace amplitudes by corresponding beam intensity amplitude at each time a reading was taken. Once this normalization is done, the DATA1 and DATA2 deflections should always be between -1 and 0.

3. Beam contrast

The DATA1 and DATA2 deflections will be related to the fringe count F(t) by

$$Y_1 = 0.5[-1+C\sin(2\pi F(t)+\theta_0)] \quad (12)$$
$$Y_2 = 0.5[-1+C\sin(2\pi F(t)+ \theta_0+\varphi)] \quad (13)$$

$\theta_0$ is the initial phase at F(t)=0, and $\varphi$ is the phase difference between $Y_1$ and $Y_2$. C=C(t) is the beam contrast, given by

$$C(t) = (Y_{max}-Y_{min})/(Y_{max} + Y_{min}) \quad (14)$$

From equations (12-13), one can obtain beam contrast C(t) from the recorded signals of $Y_1$ and $Y_2$

$$C(t) = \{(2\,Y_1+1)^2 + [((2\,Y_2+1)-(2\,Y_1+1)\cos\varphi)/\sin\varphi]^2\}^{1/2} \quad (15)$$

Note: equation (3.3) in Ref. [1] is incorrect.

4. Establishment of the fringe frequency F(t).

To determine F(t), one has to solve the inverse sine function. Even one has the definite Y and C(t), one can't get single F(t). Great caution must be taken to treat this. The following logic was used to determine the F(t):

$$\theta = \sin^{-1}((2\,Y_1+1)/C) \quad (16)$$

If one knows Y and C, one can determine $\theta$. However, $\theta$ has the multiple values. It can be $\theta=\theta_r(-\pi/2<\theta_r<\pi/2)$ or $\theta=\pi-\theta_r$. For example, if $\theta_r=-\pi/4$, $\sin\theta_r = \sin(\pi-\theta_r)=-0.707$. The first logic was designed to check $\theta$ being $\theta_r$ or $\pi-\theta_r$. This step was done by using the second measured signal $Y_2$. We assumed that $\theta=\theta_r$ and $\theta=\pi-\theta_r$, respectively and test them in equation (13) to find which closes to the experimental value $Y_2$.

As expressed in equation (16), $\theta$ is a function of time since $Y_1$ and C are the function of time (t). t increase continually, however, $\theta$ changes only between -1 and 1. This results the second testing logic. If $\theta(t_1) - \theta(t_2)$ is greater then $\pi$, we say that there is one fringe change: m=m+1 or m=m-1.

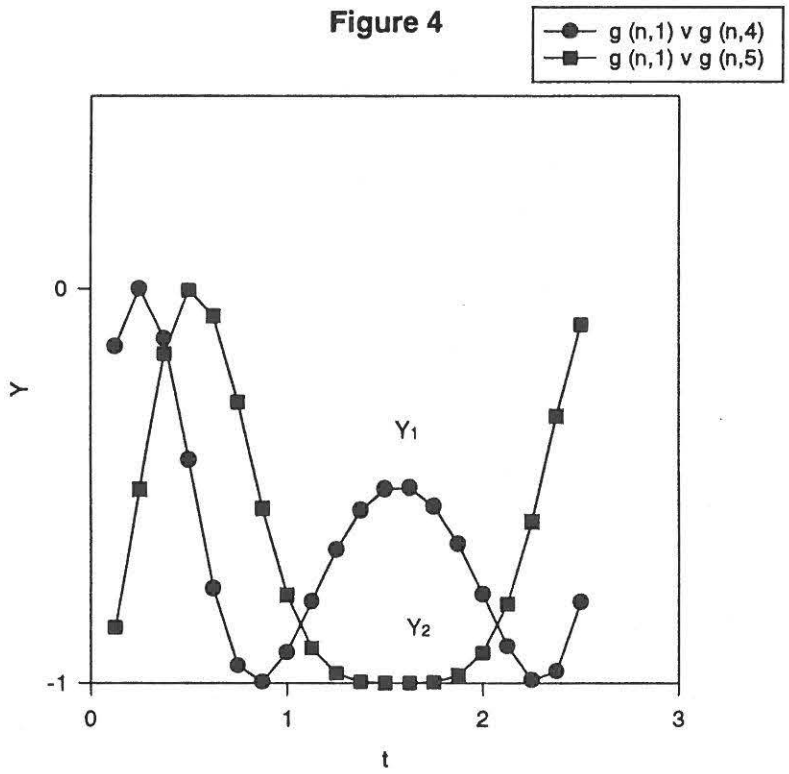
$$\theta = 2\pi F(t)+\theta_0 +2\pi m \quad (17)$$

The logic has been tested by a simple function. In order to test the logic in the ranges of increasing and decreasing velocities, we chose F(t)=sin(t). Suppose the two signals are:

$$Y_1 = 0.5[-1+\sin(2\pi(\sin(t))] \quad (18)$$
$$Y_2 = 0.5[-1+\sin(2\pi\sin(t)-\pi/2)] \quad (19)$$

here we assumed that $\theta_0=0$ and C=1. The signals $Y_1$ and $Y_2$ are expressed in Figure 4.

**Figure 4**




The following program was designed to test the logic. Since we know the answer of the F(t)=sin(t), we can easily know the correctness of the logic.

```
c   This program was designed to test the logic of solving F(t)
    dimension y1(20),y2(20),thet(20),g(20,5),f(20)
     pi=3.1415926
     do 10 n=1,20
     t=float(n)/8.
     t=sin(t)
```

```
      y1(n)=0.5*(-1.+sin(2.*pi*t))
      y2(n)=0.5*(-1.+sin(2.*pi*t-pi/2.))
   10  continue
      thetold=0.
      m=0
      do 11 j=1,20
      thet(j)=asin(2.*y1(j)+1.)
      y2ck=0.5*(-1.+sin(thet(j)-pi/2.))
      y2ck1=0.5*(-1.+sin(pi-thet(j)-pi/2.))
      diff=abs(y2ck-y2(j))
      diff1=abs(y2ck1-y2(j))
      if(diff.gt.diff1) thet(j)=pi-thet(j)
      if((thet(j)-thetold).gt.pi) m=m-1
      if((thet(j)-thetold).lt.-pi) m=m+1
      f(j)=2.*float(m)*pi+thet(j)
      thetold=thet(j)
   11  continue
      do 15 n=1,20
      g(n,1)=float(n)/8.
      g(n,2)=2.*pi*sin(g(n,1))
      g(n,3)=f(n)
      g(n,4)=y1(n)
      g(n,5)=y2(n)
   15  continue
      open(unit=2,file='g1',status='unknown')
      do 12 n=1,20
      write(2,20)(g(n,j),j=1,5)
   12  continue
      close(2)
   20  format(5(e12.5))
      stop
      end
```

To run this program, one does not need the input file. The output file of the program is g(n,5) in which g(n,1) is t; g(n,2) is F(t) calculated from the analytical formula; g(n,3) is the F(t) calculated from the logic program; g(n,4) and g(n,5) are $Y_1$ and $Y_2$, respectively. It is shown from the Table 1 that g(n,2) are exactly equal to g(n,3), regarding the correct logic.

Table 1.

| g(n,1) | g(n,2) | g(n,3) | g(n,4) | g(n,5) |
|---|---|---|---|---|
| 0.125 | 0.78335 | 0.78335 | -0.14717 | -0.85428 |
| 0.25 | 1.5545 | 1.5545 | -6.7E-05 | -0.50816 |
| 0.375 | 2.3014 | 2.3014 | -0.1276 | -0.16636 |
| 0.5 | 3.0123 | 3.0123 | -0.43554 | -0.00417 |

| | | | | |
|---|---|---|---|---|
| 0.625 | 3.6763 | 3.6763 | -0.75478 | -0.06979 |
| 0.75 | 4.2829 | 4.2829 | -0.95458 | -0.29178 |
| 0.875 | 4.8226 | 4.8226 | -0.99697 | -0.555 |
| 1 | 5.2871 | 5.2871 | -0.91967 | -0.7718 |
| 1.125 | 5.6691 | 5.6691 | -0.7881 | -0.90865 |
| 1.25 | 5.9626 | 5.9626 | -0.65754 | -0.97453 |
| 1.375 | 6.1631 | 6.1631 | -0.55988 | -0.9964 |
| 1.5 | 6.2674 | 6.2674 | -0.50787 | -0.99994 |
| 1.625 | 6.274 | 6.274 | -0.50461 | -0.99998 |
| 1.75 | 6.1826 | 6.1826 | -0.55022 | -0.99747 |
| 1.875 | 5.9947 | 5.9947 | -0.64225 | -0.97934 |
| 2 | 5.7133 | 5.7133 | -0.76977 | -0.92098 |
| 2.125 | 5.3427 | 5.3427 | -0.90392 | -0.7947 |
| 2.25 | 4.8888 | 4.8888 | -0.99224 | -0.58774 |
| 2.375 | 4.3586 | 4.3586 | -0.96902 | -0.32675 |
| 2.5 | 3.7603 | 3.7603 | -0.79 | -0.09269 |

## 5. Determination of the particle velocity

Whenever F(t) is determined by equation (17), particle velocity can be easily obtained by:

$$u(t-\tau/2)=\lambda F(t)/(2\tau(1+\delta)(1+\Delta\nu/\nu_0)) \qquad (20)$$

here $\lambda$, $\tau$, $\delta$ and $\Delta\nu/\nu_0$ are all known.

Supposed we have two normoalized signals, The particle velocity can be determined by the following program:

```
c   This program was designed for solving particle velocity from the recorded
c   fringes
    implicit real*8 (a-h,o-z), integer (i-n)
    common /list1/ signal(10001,3),npsig
    common /list5/ vel(10001,2)
    common /list6/ f(10001,2)
    common /list7/ ctrast(10001)
    integer npsig,error
    open(unit=1,file='signals.dat',status='old',iostat=error)
     if(error.gt.0) then
     write(*,*)'Unable open file'
     endif
     npsig=1201
     do i=1,npsig-1
```

```
      read(1,*) a,b,c
      signal(i,1)=a
   signal(i,2)=b
   signal(i,3)=c
     enddo
     close(1)
   call contrast(phi)
   call velocity(frconst,nwin,phi)
   open(unit=2,file='velocity.dat',status='unknown')
     open(unit=3,file='contrast.dat',status='unknown')
   do i=1,npsig-1
   write(2,10) (vel(i,j),j=1,2)
     enddo
     write(3,11)(ctrast(i),i=1,npsig)
     close(2)
     close(3)
 10    format(e12.6,',',e12.6)
 11   format(e12.6)
  stop
  end
c
c
    subroutine contrast(phi)
    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)
```

```
      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)? '
     c     //1x,' The length out of etalon in delay leg(inch) '
     c      /1x,' The length of the no-delay leg (inch)'
     c      //)
      read(*,48)etal,dx,h
```

```
      etal=etal*25.4d0
      dx=dx*25.4d0
      h=h*25.4d0
 48   format(3(f6.3))
      write(*,10)
c
      tau=2.*etal*refind/speed+2.*dx/speed
      tau=tau-2.*h/speed
c     Here tau is in microsecond
      frconst=wavel/(2.0d0*tau)
c      Here frconst is in mm/us
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
      do 60 i = 1, npsig-1

 100  if (thet0.eq.0.0) then
     phase1=dasin((2.0d0*signal(i,2)+1.0d0)/ctrast(i))
     y2try=0.5*(-1.0d0+ctrast(i)*dsin(phase1+phi))
     y2trymin=0.5*(-1.0d0+ctrast(i)*dsin(pi-phase1+phi))
     diftry=dabs(y2try-signal(i,3))
     diftrymin=dabs(y2trymin-signal(i,3))
     if (diftry.gt.diftrymin) phase1=pi-phase1
     thet0=phase1
     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.gt.diftrymin) thetr=pi-thetr
      thetl=thetr
      diff=(thetl-thetlold)
      if (diff.gt.pi) n=n-1
      if (diff.lt.-pi) n=n+1
      f(i,2)=dfloat(n)+thetl/(2.d0*pi)
      thetlold=thetl
      f(i,1)=signal(i,1)-(tau*1.0d-6)/2.0d0
c     Here f(i,1) is in second
      vel(i,1)=f(i,1)
      ff = f(i,2)
      call window(veloc,ff,frconst,nwin,delfreq,flag)
      vel(i,2)=(frconst*ff)/(1.0d0+delfreq)
c     Here velocity is in mm/us
60    continue
      return
      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
      nwarn=0
      velcurr=veloc
       if (nwin.eq.4) go to 150
  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
```

To run this program, one needs a input file signals(i,3). Signals(i,1), signals(i,2) and signals(i,3) are time, DATA1 and DATA2, respectively.

6. Modifications of the data reduction program in Ref. [1].

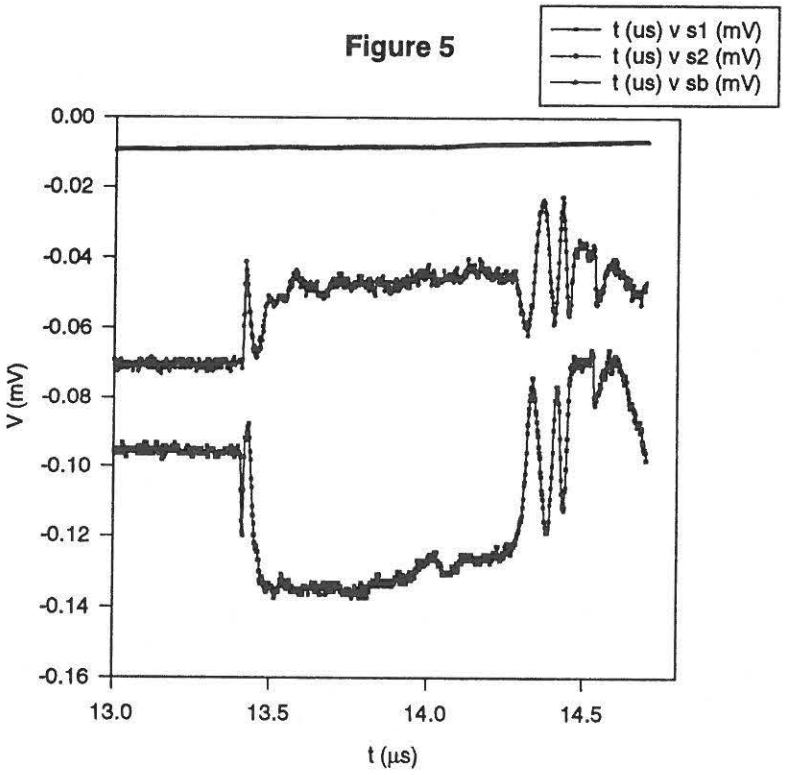
The data reduction program in [1] was designed very easily for the user. The parts of calibration of the phase shift and beam normalization of the program are correct. However, there are some mistakes in the key part of determine of velocity (velocity subroutine). So the subroutines of beam contrast and velocity have been rewritten as described above. These new subroutines have been put into the old program. The all operation procedures are the same to those described in [1]. User can follow the descriptions in [1]. Several things must be taken care. 1) In the Zero Adjustments, the zero of the beam intensity has to be adjusted manually. If one chooses automatically adjustment, zero may appear in BIM, which will give unreasonable results of normalization of DATA1 and DATA2. 2) The velocity from the calculation refers the velocity changes (change of the fringes). Sometimes, offset of the velocity may appear. This dose not effect the correctness of the results.

Some corrections have also been made in the format of the input and output files. Now the all operations are reasonable to the PC computer.

The revised program has been given in the appendix.

An example was given as the following. The measured DATA1, DATA2 and BIM are shown in Figure 5. The velocity obtained from the program was given in Figure 6.

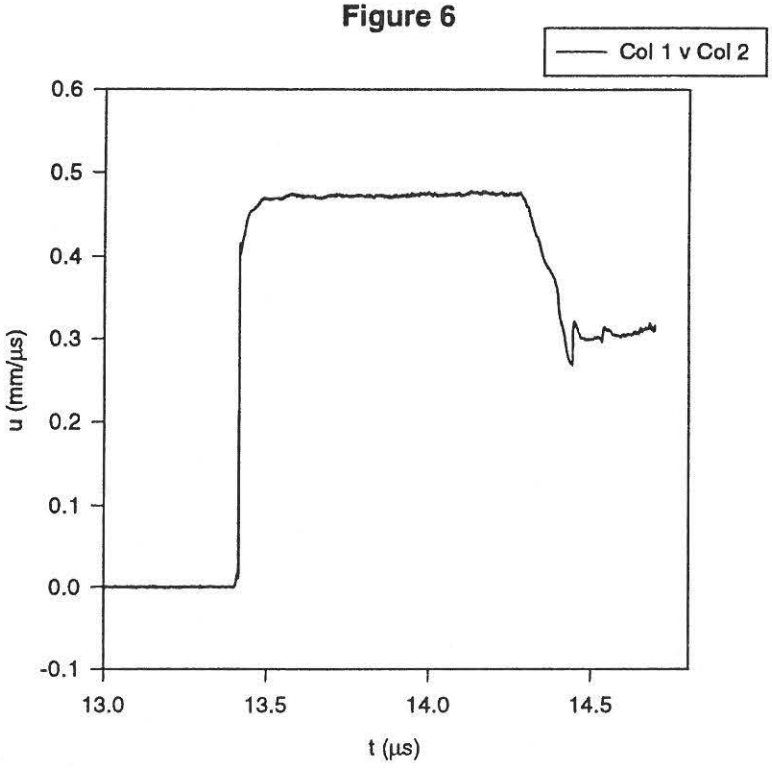
Figure 5




7. Predication of fringes from the particle velocity

For designing the experiment, one has the particle velocity profile from the simulation and wishes to know the corresponding fringes of the VISAR measurements. This is a inverse problem of we did above. Consider of equations (8,12,13), we can get F(t) from u(t). $\theta_0$ and $\varphi$ can be assumed. We can predict the signal profiles $Y_1$ and $Y_2$ From (12 and 13). Solution of this problem is not straightforward since we do not know the beam contrast C(t). The method we used here is a feedback treatment. First, we assumed C=1 and calculate the signals $Y_1$ and $Y_2$. After we have $Y_1$ and $Y_2$, we can calculate a new C(t) from equation (15). Then we can calculate the new $Y_1$ and $Y_2$ from (12 and 13). These procedures are repeated. It is found that C(t) will be stable after cyclic is greater than 1. The program is given as following:

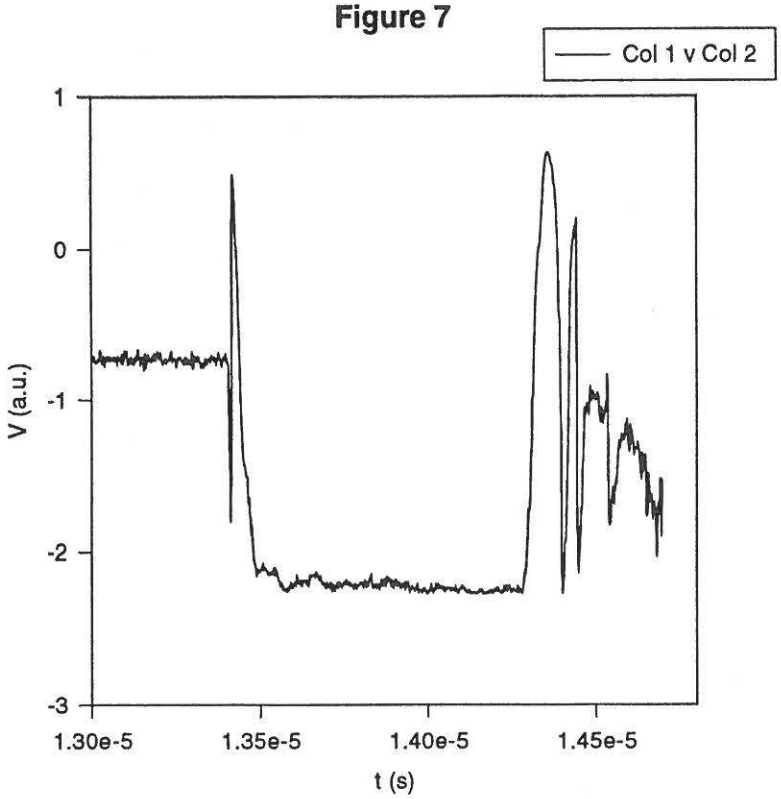
**Figure 6**




```
c   This program was designed to predict the fringes from the velocity profile.
    implicit real*8 (a-h,o-z), integer (i-n)
    dimension signal(10001,3)
    dimension vel(10001,2)
    integer npsig,error
    open(unit=1,file='velocity.dat',status='old',iostat=error)
     if(error.gt.0) then
     write(*,*)'Unable open file'
     endif
     npsig=849
     do i=1,npsig
      read(1,*) a,b
      vel(i,1)=a
     vel(i,2)=b
     enddo
     close(1)
     do 60 i=1,npsig
  ff=vel(i,2)/0.147
  thet=ff*2.0*3.1315926
  signal(i,1)=vel(i,1)
  signal(i,2)=0.5*(-1.+sin(thet))
```

```
pil=-85./180.*3.14159
signal(i,3)=0.5*(-1.+sin(thet+pil))
c1=signal(i,2)
c2=signal(i,3)
   c=(2.*c1-1.)**2+((2.*c2-1.-(2.*c1-1.)*cos(pil))/sin(pil))**2
   c=sqrt(c)
   signal(i,2)=0.5*(-1.+c*sin(thet))
60  continue
open(unit=2,file='signal.dat',status='unknown')
   do i=1,npsig
write(2,10) (signal(i,j),j=1,2)
   enddo
   close(2)
10  format(e12.6,',',e12.6)
11  format(e12.6)
stop
end
```

**Figure 7**



To run this program, one needs an input file of velocity.dat. The output file will be signal.dat. Figure 7 is a predicted fringe from the velocity profile shown in Figure 6. It is shown that the predicted signal is definitely the same to the measured signal shown in figure 5.

VI Summary described

We ~~deiseeibed~~ the details of operation of SDC VISAR. User can use these information combined with Ref. [1] to operate the VISAR.

References

[1] G.F. Raiser, Background and data reduction on the SDC VISAR, internal report SDC, October 1994.
[2] L.M. Barker, Exp. Mech. 12, 40(1972).

# Appendix: Data reduction program

```
c    program sdevis
c
c    Latest Revision: 10/29/94 (G.F. Raiser)
c
c
c    Changed Version: 5/19/95 by G. Yuan
c
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                                       corrections, 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,'*  Program was revised by G. Yuan 5/19/95   *'
     c /11x,'*                                           *'
     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-4) 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)? '
     c      /1x,' The length out of etalon in the delay leg(inch)'
     c      /1x,' The length of the no-delay leg (inch)'
     c      //)
       read(*,48)etal,dx,h
       etal=etal*25.4d0
       dx=dx*25.4d0
       h=h*25.4d0
 48    format(3(f6.3))
       write(*,10)
c
       tau=2.*etal*refind/speed+2.*dx/speed
       tau=tau-2.*h/speed
c      Here tau is in microsecond
       frconst=wavel/(2.0d0*tau)
c      Here frconst is in mm/us
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))
              y2try=0.5*(-1.0d0+ctrast(i)*dsin(phase1+phi))
              y2trymin=0.5*(-1.0d0+ctrast(i)*dsin(pi-phase1+phi))
              diftry=dabs(y2try-signal(i,3))
              diftrymin=dabs(y2trymin-signal(i,3))
              if (diftry.gt.diftrymin) phase1=pi-phase1
              thet0=phase1
              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.gt.diftrymin) thetr=pi-thetr
      thetl=thetr
      diff=(thetl-thetlold)
      if (diff.gt.pi) n=n-1
      if (diff.lt.-pi) n=n+1
      f(i,2)=dfloat(n)+thetl/(2.d0*pi)
      thetlold=thetl
      f(i,1)=signal(i,1)-(tau*1.0d-6)/2.0d0
      vel(i,1)=f(i,1)
c     Here velocity is in second
      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
      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(3(e12.5,','))
 20   format(e12.5,',',e12.5)
      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(e12.5,',',e12.5)
      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(e12.5)
      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(e12.5,',',e12.5)
      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
```