A 3-COMPONENT LASER-DOPPLER VELOCIMETER
DATA ACQUISITION AND REDUCTION SYSTEM

BY

L. C. Rodman, J. H. Bell and R. D. Mehta

Stanford University
Department of Aeronautics and Astronautics
Stanford, CA  94305

AUGUST 1985
A 3-COMPONENT LASER-DOPPLER VELOCIMETER
DATA ACQUISITION AND REDUCTION SYSTEM

BY

L. C. Rodman, J. H. Bell and R. D. Mehta

The work here presented has been supported by NASA Grant NCC 2-294.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>iv</td>
</tr>
<tr>
<td>NOMENCLATURE</td>
<td>v</td>
</tr>
<tr>
<td>1. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. OPTICAL SYSTEM AND SIGNAL PROCESSING HARDWARE</td>
<td>3</td>
</tr>
<tr>
<td>2.1 Optics Table</td>
<td>4</td>
</tr>
<tr>
<td>2.2 Transmitting Optics</td>
<td>5</td>
</tr>
<tr>
<td>2.3 Receiving Optics</td>
<td>6</td>
</tr>
<tr>
<td>2.4 Signal Processing Hardware</td>
<td>6</td>
</tr>
<tr>
<td>2.5 Computer Interface</td>
<td>7</td>
</tr>
<tr>
<td>2.6 HP 9845A Desk-Top Computer</td>
<td>8</td>
</tr>
<tr>
<td>3. DATA ACQUISITION AND REDUCTION SOFTWARE</td>
<td>8</td>
</tr>
<tr>
<td>3.1 Data Acquisition Software</td>
<td>9</td>
</tr>
<tr>
<td>3.2 Data Reduction Software</td>
<td>10</td>
</tr>
<tr>
<td>4. OPERATING PROCEDURES</td>
<td>15</td>
</tr>
<tr>
<td>4.1 Alignment Procedures</td>
<td>15</td>
</tr>
<tr>
<td>4.2 Signal Processing</td>
<td>17</td>
</tr>
<tr>
<td>5. INHERENT PROBLEMS</td>
<td>19</td>
</tr>
<tr>
<td>6. SAMPLE RESULTS</td>
<td>21</td>
</tr>
<tr>
<td>7. CONCLUDING REMARKS</td>
<td>22</td>
</tr>
<tr>
<td>APPENDIX -- SOFTWARE FOR THE HP 9845B COMPUTER</td>
<td>23</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>67</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>68</td>
</tr>
<tr>
<td>TABLE 1</td>
<td>69</td>
</tr>
<tr>
<td>FIGURES</td>
<td>70</td>
</tr>
</tbody>
</table>
ABSTRACT

This report describes a Laser Doppler Velocimeter capable of measuring all three components of velocity simultaneously in low-speed flows. All the mean velocities, Reynolds stresses, and higher-order products can then be evaluated. The approach followed is to split one of the two colors used in a 2-D system, thus creating a third set of beams which is then focused in the flow from an off-axis direction. The third velocity component is computed from the known geometry of the system. In this report, the laser optical hardware and the data acquisition electronics are described in detail. In addition, full operating procedures and listings of the software (written in BASIC and ASSEMBLY languages) are also included. Some typical measurements obtained with this system in a vortex/mixing layer interaction are presented and compared directly to those obtained with a cross-wire system. A brief description of the present system together with a review of existing 3-D Laser Doppler Velocimeters is given in Ref. 1.
<table>
<thead>
<tr>
<th>Fig.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Schematic of the optics table layout.</td>
</tr>
<tr>
<td>2</td>
<td>Schematic of the transmitting optics.</td>
</tr>
<tr>
<td>3</td>
<td>Details of the probe volumes in the 3-D system.</td>
</tr>
<tr>
<td>4</td>
<td>Schematic of the receiving optics.</td>
</tr>
<tr>
<td>5</td>
<td>Evaluation of the effective probe length.</td>
</tr>
<tr>
<td>6</td>
<td>Schematic of the signal processing electronics.</td>
</tr>
<tr>
<td>7</td>
<td>Computer interface simplified block diagram.</td>
</tr>
<tr>
<td>8</td>
<td>Block diagram for data acquisition program.</td>
</tr>
<tr>
<td>9</td>
<td>Block diagram for data reduction program.</td>
</tr>
<tr>
<td>10</td>
<td>NASA LDV-A/D computer interface connections and settings.</td>
</tr>
</tbody>
</table>
| 11   | Experimental rig.  
|      | (a) Overall schematic  
|      | (b) Details of boundary layer trips and coordinate system |
| 12   | Secondary velocity plots.  
|      | (a) Cross-wire measurements  
|      | (b) LDV measurements |
| 13   | Comparison of X-wire and LDV measurements in a vortex/mixing layer interaction. |
NOMENCLATURE

a(i)  
ith sample value

d_f  
fringe spacing

f  
frequency

i  
sample number

Ns  
number of samples

R  
range setting on Macrodyne processor

S<a(i)>  
sum of a(i) from i = 1 to i = Ns

Sa  
average of quantity a, defined by equations (3)

u, v, w  
mean velocity components in x, y, z directions, respectively

u, v, w  
instantaneous velocity components in x, y, z directions, respectively

U_o  
U_1 - U_2, velocity difference between the two streams in the mixing layer

x, y, z  
Streamwise, normal, and spanwise coordinate directions, respectively

Subscripts:

bragg  
Bragg

mix  
mixing

dopp  
doppler

res  
resultant

1  
quantity measured in high-speed side of mixing layer

2  
quantity measured in low-speed side of mixing layer

Superscripts:

'  
fluctuating quantity, e.g., u = \bar{u} + u'

-  
time average
1. INTRODUCTION

Our ability to understand and model turbulent flows still relies heavily on the availability of accurate measurements of mean and fluctuating quantities within the flow. Until recently, the hot wire was the only reliable tool available for the measurement of fluctuating velocities. In fact, almost all of our present knowledge about turbulent flows is based on measurements made with hot wires. In relatively simple flows (moderately two-dimensional with small cross-flows), reliable and accurate hot wire measurements are now possible with fully automated data acquisition and reduction systems which minimize errors due to drifts in calibrations. An example of such a system is given in Ref. 2.

However, as we turn our attention towards more complex turbulent flows, a need for more sophisticated measurement techniques has become apparent. These complex flows include those with compressibility effects, strong three-dimensionality (with steep mean gradients), flow reversals, and time-dependent behavior. Since about the mid-sixties, the most popular alternative tool for measuring mean and fluctuating velocities in turbulent flows has been the Laser Doppler Velocimeter (LDV).

The most popular LDV arrangement used for wind tunnel measurements is the dual beam or fringe method. In this method, one of the laser lines is split into two lines of equal intensity which are then focused through a lens so that they cross over at the focal point. The flow is seeded with small particles (typically less than 3-4 μm in diameter) which follow the fluid motion. As these particles pass through interferometric fringes created by the crossed laser beams, light is scattered off them which is received by a photodetector. The frequency of this scattered light, along with a knowledge of the fringe pattern formed by the laser beams, provide the means to calculate the velocity of the particle. The fringe method, especially in
the forward scatter, off-axis mode, generally offers the best signal-to-noise ratios and spatial resolution.

Although LDV systems are somewhat complex and tiresome to set up, they have certain advantages over hot wires for turbulence measurements. The fact that Laser Doppler Velocimetry is non-intrusive is especially beneficial in the measurement of unstable flow phenomena which are very sensitive to the presence of measurement probes. In certain situations, LDV systems can also provide greater spatial resolution and better directional discrimination than hot wires. This makes it possible to use LDV systems for the measurement of separated flows. Since an LDV measures the velocity directly, independent of the thermodynamic properties of the flow, it is particularly attractive for velocity measurements in compressible flows. Furthermore, the calibration converting the frequency to velocity is linear and easy to implement in software. This feature also allows for uniform sensitivity in measuring both moderate and high turbulence intensities.

Two-color LDV systems capable of measuring two components of velocity simultaneously are now being widely used. However, the main interest in the present investigation was to study three-dimensional interactions where it is desirable to obtain measurements of all three velocities. Hence, the first objective was to develop a laser velocimeter system capable of measuring all three components of velocity simultaneously so that all six components of Reynolds stress may be computed. Another objective was to compare these LDV measurements directly with those obtained with hot wires in flow fields where both techniques are expected to perform satisfactorily. The purpose of this is to evaluate the performance of the system quantitatively and objectively.

The approach followed in the present investigation was to convert an existing two-component LDV system into one capable of measuring all three components of velocity simultaneously. The system utilizes two wavelengths (488.0 and 514.5 nm) from a 4-watt Argon-Ion laser. The main four-beam matrix measures
u and v directly. The green line in the four-beam matrix is split (in half) using a dichroic filter and directed over the top of the traverse mechanism with mirrors, giving the third beam pair for the measurement of the w-component. This pair of beams (with rotated polarization) measures $w \sin 45^\circ + v \cos 45^\circ$. Scattered light is collected in the off-axis forward scatter mode using two collection lenses.

Signal processing is accomplished with single-particle burst counters, and the validated data are multiplexed through a "home-built" interface to an HP 9845B desk-top computer. Some selected first and second order products are reduced on-line, and the raw data is dumped onto floppy disk. An off-line program reduces the data, giving up to third order quantities and also plots histograms of the raw data for each channel. The software includes the capability to filter out noise by examining the histograms.

The optical and signal processing hardware is described in Section 2. The data acquisition and reduction software is described in Section 3 and detailed operating procedures are given in Section 4. Some problems, inherent to 3-component LDV systems, are presented in Section 5. Sample results from an experiment measuring mean and turbulence quantities in a vortex/mixing layer interaction are compared directly to results obtained using crossed hot-wire anemometry in Section 6, and concluding remarks are presented in the final section. Complete software listings written in BASIC and ASSEMBLY languages to run on the HP 9845B desk-top computer are included in the appendix.

2. OPTICAL SYSTEM AND SIGNAL PROCESSING HARDWARE

The hardware for the 3-component LDV system can be divided into three categories: the optical system, the signal processing instrumentation, and the computer. The LDV optics consist of the optics table, where the laser beam is split into green and blue beam pairs, the transmitting optics, where the beams are
directed into the flow field, and the receiving optics, where the scattered light is picked up by photodetectors. The signal processing instrumentation consists of amplifiers, filters, burst counters and a computer interface. The sampling procedures are all computer controlled.

2.1 Optics Table

The optics table consists of the laser and all the optical elements needed to provide the necessary four-beam matrix. Fig. 1 shows a schematic of the arrangement with component numbers as referred to in this section. A 4-watt Argon-Ion laser (Lexel Model 95) is used to produce the main beam. The beam then passes through a collimator (1), which ensures that the beam waist occurs at the focal point of the transmitting lens.

The collimated beam is passed through a color separator box, which consists of a polarization rotator (2), an attenuator (3), and a pair of high dispersion Brewster angle prisms (4) which are used to separate the multi-line beam into two colors, blue (488 nm) and green (514.5 nm). These two beams are then reflected across the box by mirror (5), and out of the box by mirrors (6, 7).

Following the color separation, the green beam's polarity is rotated to horizontal (8), and the beam is split into two beams in the vertical plane (9). Most beam splitters prefer this type of perpendicular polarization for maximum efficiency. Using the beam displacer (10), the blue beam is then moved to the center of the optics, and its polarity is rotated to vertical (11). The blue beams are split in the horizontal plane (12). At this point, the four beams are each displaced 25 mm from the optical axis.

Two Bragg cells (13, 14) are used to shift the frequency of one beam from each pair. The unshifted beam passes through an optical rod so that the path lengths are matched. The frequency is shifted by a fixed amount of 40 Mhz. This shift creates
a moving system of fringes at the beam intersection point, allowing for directional discrimination of the velocity. Frequency shifting also helps to reduce the percentage of frequency change in highly turbulent flow, to reduce fringe bias, and to optimize frequency, thus enabling easy removal of the pedestal by high pass filtering.

The four beams are then passed through a beam steering module (15). The module consists of a set of wedge prisms that can be independently rotated about the beam axis to steer the shifted beam in any direction. This allows a more precise alignment of the beams. The beams are finally passed through a beam displacer (16) to reduce the beam spacing to 13 mm and a rotating prism (17) before leaving the optics table. The rotating prism enables the four-beam matrix to be rotated independently so that the beams may be aligned relative to the tunnel axes.

2.2 Transmitting Optics

The transmitting optics (Fig. 2) are mounted on a traversing mechanism with three degrees of freedom. The traverses are driven by individual stepper motors. The four beams from the optics table are directed by a set of five mirrors through a dichroic filter before being focused by a 380 mm (15 inch) focal length lens. This main four-beam matrix measures u and v directly. The dichroic filter, set at an angle of about 30 degrees to the incoming beams, splits the green beams in half, which provides the third beam pair for the third velocity component, w. This third beam pair is directed over the top by mirrors, passed through a polarization rotator (giving it a different polarity than the main-axis green beam pair) and then focused at the focal point of the main beam set. Since this third beam pair intersects the main measuring volume at a $45^\circ$ angle to the main axis, it measures v and w with equal sensitivity, with the measured component being $w_v = (v \cos 45^\circ + w \sin 45^\circ)$. Since v is measured directly, w can be evaluated using the equation:
\[ w = \frac{w_v - v \cos 45^\circ}{\sin 45^\circ} \]  

Typical probe volume dimensions for each beam pair in the present configuration are 10 mm in length and 0.2 mm in diameter (Fig. 3). However, the actual "viewed" dimensions are reduced considerably, as discussed below, in Section 2.3.

2.3 Receiving Optics

The detector system (Fig. 4) is in the off-axis forward scatter mode. The receiving optics are mounted on a traversing gear, also run by stepper motors, which moves synchronously with the transmitting optics. Scattered light is collected by two 380 mm (15 inch) focal length lenses. The collimated light is passed through filters to separate the colors and the off-axis line is additionally passed through a polarization filter to avoid collecting light scattered by the main axis green pair. The collected light is then focused by 250 mm (10 inch) focal length lenses onto pin-hole apertures mounted in front of the three photomultiplier tubes. The collection angle and diameter can be adjusted to select the effective (viewed) probe length (Fig. 5). In the present set-up, a collection angle of 30 degrees and an aperture diameter of 0.5 mm were used to give an effective length of about 1.5 mm.

2.4 Signal Processing Hardware

The signals from the photomultiplier tubes are amplified and relayed to the signal processors via high-pass filters and mixers (Fig. 6). The amplifiers used are EIN model 403LA with a fixed gain of 37 dB and the filters are Allen Avionics F2440 with a fixed high-pass cutoff of 10 MHz. The filtered signals are mixed electronically with sine waves from three Tektronix SG503 Levelled Sine Wave Generators. The mixer is commercially available from Hewlett-Packard, model 10534A. The mixing procedure
is necessary for low-speed flows, where the actual Doppler frequencies are small compared to the Bragg frequency of 40 MHz. So in order to reduce the effective measured frequency and hence improve the counter resolution, the incoming signals are mixed with sine waves of known frequency.

The mixed signal is fed into single particle burst counters (Macrodyne model 2096-2 and 3003) via high-pass/low-pass filters and an amplifier (x10). The counters measure the zero crossings of the Doppler signal, which is related to the Doppler frequency by the range set on the Macrodyne. The range is set manually based on expected flow velocities, since it limits the frequencies that the processor can see for the given 10 bits of resolution.

The processors use two checks to validate a Doppler signal. The first check is the usual 5/8 comparison, where the processor checks the frequency for 5 zero crossings against that for 8 crossings. The second check is the multi-sequence check. Positive and negative thresholds are set on the signal, and a validated output is permitted only if, for all eight fringe crossings, the signal passes through a positive threshold, a zero level, and a negative threshold in the proper sequence. The digital data (consisting of a 10 bit data word with 3 bits giving the range) and a sync pulse (produced every time the front end of a valid burst is detected) are passed to the computer interface.

2.5 Computer Interface

A NASA LDV-A/D computer interface (CI) is used to transfer data from the LDV signal processor to the computer (Fig. 7). The CI can interface either digital or mixed analog and digital data to an HP 9845B desk-top computer. The CI consists of an eight-channel multiplexer, a four-channel A/D converter, and an event synchronizer with time interval counter.

For use with the LDV, the inputs to the CI are all digital. Six of the eight words come from the processor, and two are time and status words from the synchronizer. The inputs are
multiplexed to a single digital data channel output.

The CI can accept data from the three processors in either random mode or sync mode. In random mode, the CI will accept data inputs when an event occurs on any of the three channels. A dead time (between 5 and 50 ps) is set in this mode, which controls the minimum time between samples to ensure that a given particle is sampled only once. In sync mode, the CI will accept data inputs when all three processors sample simultaneous events. In this mode, a coincidence time (between 5 and 50 ps) has to be set, which determines the time window within which all three events must occur. A detailed description of the CI can be found in Ref. 5.

2.6 HP 9845B Desk-top Computer

Multiplexed data are passed to the HP computer from the NASA LDV-A/D using the HP 98032A high-speed 16-bit parallel interface. Jumpers labeled "9,B,D" are connected inside the 98032A for proper operation with the LDV-A/D. A select code of 10 (screw setting on the 98032A) is set for use with the software described in the appendix. A data buffer of 24 kbytes is provided in the memory of the HP 9845B desk-top computer for storage of up to 2000 samples obtained from one measurement location. Since three of the six data words passed for each sample are merely monitored and discarded as described above, 12 kbytes of raw data remain to be stored for each point. An HP 9895A floppy-disk drive is used for archival storage of raw data. The buffered data are written in real time to a sequential-access floppy-disk file. Enough header information is written to each file to identify the run, and to reproduce calibration tables.

3. DATA ACQUISITION AND REDUCTION SOFTWARE

Data acquisition and reduction on the HP 9845B is done via
two programs. The data acquisition program controls the processor sampling. The program accepts heading and initialization parameters provided by the operator, performs a fast I/O handshake to acquire the raw data from the LDV-A/D computer interface, and writes the raw data and initialization parameters onto floppy-disk. In addition, the data acquisition program has a limited capability for on-line data reduction so that key results may be monitored during a run. The off-line data reduction program converts the raw data into instantaneous velocities and computes all the statistical quantities. The off-line program also has options to plot histograms of the raw data, to plot profiles of the reduced data, and to filter out noise.

3.1 Data Acquisition Program

The data acquisition program structure can be divided into four areas: (1) initialization of variables, (2) data acquisition, (3) limited data reduction, and (4) raw data storage (see the block diagram in Fig. 8). During the initialization stage of the program, relevant test parameters are input to be saved along with the data. These parameters include the spatial coordinates, Bragg and mixing frequencies, and angle of the third beam pair. Fringe spacings and probe dimensions are also calculated at this time.

Data is acquired from the LDV computer interface by means of a fast I/O handshake. The operator requests that a certain number of data samples be taken and stored in the data buffer as described in Section 2.6. The 16-bit data words are put into an integer array, three words wide, containing values for the three velocity components. The number acquired from the counter occupies only bits 0-9 of the data word. Bits 10-13 specify the range, and bits 14 and 15 are unused. The data array, along with the initialization parameters given by the operator, are stored in a floppy-disk file. As an option, some of the data can be reduced on-line, before the raw data is written.
to floppy-disk. This on-line reduction feature uses the same algorithms as the off-line data reduction program, but is faster since it computes fewer quantities. Data acquisition times depend on many factors other than the software, but on-line data reduction takes about 30 seconds per point on the HP 9845B, and each file write to the floppy-disk requires about 20 seconds, assuming 2000 samples are taken per point.

3.2 Data Reduction Program

A separate off-line program is used to reduce the complete data. The data reduction program has four major sections: (1) data read from floppy-disk, (2) conversion of raw data to instantaneous velocities, (3) calculation of sums of instantaneous velocity components, and (4) calculation of moments from the sums (see the block diagram in Fig. 9). Since the HP 9845B is a relatively slow micro-computer, the bulk of the calculation routines are written in ASSEMBLY language to reduce computation time.

The program reads the initialization parameters and raw data which were written on floppy-disk by the data acquisition program. The raw data is used together with the mixing frequencies, Bragg shift frequency, third beam crossing angle, and fringe spacings to calculate the three velocity components for each sample. The velocities for each sample are calculated in turn, and a running sum is maintained. After the samples have been summed, the average values of the various moments are computed, and the mean velocities and turbulence quantities are calculated from them. Reading from the floppy-disk takes about 20 seconds, while the data reduction including histogram plots requires about 90 seconds per point on the HP 9845B.

The procedure for data reduction is fairly straightforward. The raw data is converted to frequencies using the range set on the processor. The measured frequency is given by the relation
\[ f_{\text{res}} = \frac{3.2 \times 10^4}{2R \times D} \]  

where \( D \) = raw data.

The velocities are then calculated from the resultant frequencies, by the relation:

\[ \text{velocity} = d_f \times (f_{\text{bragg}} - f_{\text{mix}}) - f_{\text{res}} \]  

Once the velocities have been evaluated, average values of the various moments are computed as defined below:

\[
\begin{align*}
Su &= \frac{S<u(i)>}{Ns} \\
Sv &= \frac{S<v(i)>}{Ns} \\
Sw &= \frac{S<w(i)>}{Ns} \\
Suu &= \frac{S<u(i)u(i)>}{Ns} \\
Suv &= \frac{S<u(i)v(i)>}{Ns} \\
Swv &= \frac{S<v(i)w(i)>}{Ns} \\
Suuw &= \frac{S<u(i)w(i)>}{Ns} \\
Suw &= \frac{S<u(i)w(i)>}{Ns} \\
Svv &= \frac{S<v(i)v(i)>}{Ns} \\
Svw &= \frac{S<v(i)w(i)>}{Ns} \\
Sww &= \frac{S<w(i)w(i)>}{Ns} \\
Suuv &= \frac{S<u(i)u(i)v(i)>}{Ns} \\
Suuw &= \frac{S<u(i)u(i)w(i)>}{Ns} \\
Suvv &= \frac{S<u(i)v(i)v(i)>}{Ns} \\
Suww &= \frac{S<u(i)w(i)w(i)>}{Ns} \\
Suvw &= \frac{S<u(i)v(i)w(i)>}{Ns}
\end{align*}
\]  

Using these definitions, the signal statistics are then calculated assuming nearly infinite sample size:

\[
\begin{align*}
\overline{u} &= Su \\
\overline{v} &= Sv \\
\overline{w} &= Sw \\
\overline{u^2} &= Suu - SuSu \\
\overline{v^2} &= Svv - SvSv
\end{align*}
\]
\[
\begin{align*}
\overline{w'^2} &= Sw - SwSw \\
\overline{u'v'} &= Suv - SuSv \\
\overline{u'w'} &= Suw - Susw \\
\overline{v'w'} &= Svw - SvSw \\
\overline{u'^2v'} &= Suuv - 2SuSuv - SvSuu + 2SvSuSu \\
\overline{u'^2w'} &= Suuw - 2SuSuw - SwSuu + 2SwSuSu \\
\overline{u'v'^2} &= Suvv - 2SvSuv - SuSvv + 2SuSvSv \\
\overline{u'w'^2} &= Suww - 2SwSuw - SuSww + 2SuSwSw \\
\overline{u'v'w'} &= Suvw - SUSvw - SvSuw - SwSuv + 2SuSvSw
\end{align*}
\]

The implementation of the data reduction software is somewhat more complex. In order to achieve reasonable running times, the ASSEMBLY code is optimized for speed rather than clarity of operation. Operations which do not change between samples are performed only once. Thus,

\[
\text{velocity} = df \ast ((f_{\text{bragg}} - f_{\text{mix}}) - f_{\text{res}}) \ast 10^6 \quad (6)
\]

becomes

\[
f_{\text{int}} = f_{\text{bragg}} - f_{\text{mix}} ; \quad d_{\text{int}} = df \ast 10^6 \quad (7)
\]

hence,

\[
\text{velocity} = d_{\text{int}} \ast (f_{\text{int}} - f_{\text{res}}) \quad (8)
\]

Additional calculations are required to obtain the \( w \) component velocity. Since the third set of beams measures \( w_v = w \sin 45^\circ + v \cos 45^\circ \), to find \( w \), we must also perform the calculation:

\[
w = \left( w_v - v \cos 45^\circ \right) / \sin 45^\circ \quad (9)
\]

The raw data output of the digitizer is in the form of a 10 bit data word, giving a range of possible values (counts) from 0 to 1023. Since 2000 samples are normally taken for each point, a given value may be encountered many times. Accordingly, each time a new value of the raw data is encountered, the corresponding velocity is calculated and stored in a look-up table. The next time that value is encountered by the program, the
proper velocity can be easily looked up, eliminating the need for another time-consuming real variable calculation. The exact running procedure of the data reduction program therefore, is as follows:

1. Reads header and all raw data for a particular point from floppy disk.

2. Strips bits 10-13 from three raw data words corresponding to the three different velocity components, and uses them to calculate the three ranges set on the A/D.

3. Calculates various intermediate values which remain the same throughout the point.

4. Reads a data word, strips off bits 0-9, and uses this raw datum as an index to look up its corresponding velocity.

5. If it is a new value, the program calculates the velocity using the raw datum, and stores it in the appropriate place in the look-up table.

6. For the w velocity component, it finds the actual velocity from \( w = (w_v - v \cos 45^\circ) / \sin 45^\circ \).

7. Performs steps 4, 5, and 6 three times—once for each velocity component of the sample.

8. Updates the running sums of the velocity components and products of the velocity components.

9. Performs steps 4 through 8 for all samples.

10. Uses the sums to obtain the average velocities, Reynolds stresses, and third order products, and then prints these quantities.
11. Plots histograms of the raw data for all three channels.

12. Performs steps 1 through 11 for each profile point.

13. Plots profiles of the reduced data.

14. Tabulates normalized data profiles.

15. Writes a summary file containing the reduced data to disk.

The data reduction program also has a routine to filter noise and spurious data. Each of the 2000 data samples has 3 counts corresponding to the 3 velocities associated with it. The filtering routine causes the data reduction program to ignore samples associated with counts which are excessively far from the mean.

The filtering routine first sorts the data into three frequency tables, one for each channel. In a table, each count, \( i \), has associated with it a number, \( S_i \), which is the number of samples with that particular count.

The filtering routine finds the average count for each channel by going through the frequency tables and using the formula

\[
C_j = \frac{\sum_{i=1}^{1024} i \times S_{ij}}{N_s}
\]  

(10)

where \( j = 1,2,3 \) = channel number

\( C_j \) = average count

\( N_s \) = number of samples.

Next, the filtering routine finds the standard deviation by performing the summation

\[
\sigma_j^2 = \frac{\sum_{i=1}^{1024} S_{ij}(i - C_j)^2}{N_s}
\]  

(11)
where $\sigma_j = \text{standard deviation.}$

An input variable called $S_{\text{dev}}$ is read by the filtering routine. All counts further than $S_{\text{dev}}$ standard deviations away from the mean will be filtered out. This is done by multiplying $S_{\text{dev}}$ by $\sigma_j$ for each channel, and going through the frequency tables one more time. If the magnitude of $(i - C_j) > S_{\text{dev}} \times \sigma_j$, then $S_{ij}$ is set equal to zero.

The routine which calculates the statistical quantities takes each sample one at a time. The three counts associated with each sample are found and looked up in the frequency table. If for any count the table entry ($S_{ij}$) is zero, then the sample is discarded. Thus any sample which is excessively far from the mean in any one of its three counts is not used.

4. OPERATING PROCEDURE

4.1 Alignment Procedure

To achieve the best possible beam crossing and the most effective measuring volume, each module in the optics system must be carefully aligned. Detailed alignment instructions for individual optical components are given in Ref. 3. The overall alignment procedure is described in this report. Component numbers in this section refer to those shown in Fig. 1.

The first step is to check that the laser output beam is parallel to the optics table at the specified height, using the system alignment blocks. The collimator (1) should be positioned so that the laser beam goes through the center of the lens, and focused so that the beam waist occurs at the cross-over point. The aligned beam then passes through the polarization rotator (2) and into the color separator box. The attenuator (3) ensures that the beam has horizontal polarity at this point.

Each component of the color separator must be aligned separately. The dispersion prism (4) should be aligned so that the path length is the same in both prisms and is parallel to the
base of each prism. Several beams emerge from the prism, with the two brightest ones being green (514.5 nm) and blue (488 nm). The two beams are reflected out of the box using mirrors. The beams should be centered on mirrors (6) and (7). As the beams come out of the box, alignment blocks are used to check the beam positions. If both beams are off-axis in the same direction, mirror (5) is used to align them. If only one beam is off, the appropriate mirror, (6) or (7), is used for the adjustment. Beam splitter efficiency is maximized when the beam polarization is perpendicular to the plane of the split beams. This is achieved through components (8) and (11). The beam splitters (9) and (12) cannot split the beam intensity exactly in half, so one beam of each pair is always slightly brighter. Since the Bragg cells (13) and (14) normally attenuate the shifted beam, the brightest beam is frequency shifted so that the output beam pairs have nearly equal intensities. Detailed instructions for aligning the Bragg cells are given in Ref. 4. After passing through the Bragg cells, directional wedges are used to project the beams onto a distant surface (~ 3 m). Any beam misalignments are more easily seen this way, and with the use of a marked mask, the beams can be adjusted to the correct orientation. Mirrors (6, 7) are used to adjust the unshifted beams, and the beam steering modules (15) are used for rotating the shifted beams along two circular arcs.

The next check is to ensure that the beams are parallel to each leg of the traverse mechanism on the transmitting optics table, so that beam alignment is maintained while traversing. The dichroic filter (Fig. 2) is adjusted so that the green beam pairs are split equally. Once the beams pass through the transmitting optics, the four beams must be arranged so that they all cross at the same position. A microscope objective is used to view the beam crossing. If the four beams are not symmetric about the optical axis, mirrors (6) or (7) can be adjusted to correct this. The beam steering modules (15) are used to ensure that the beams cross at the same point along the axis. The
third pair of beams are now aligned (by eye) so that they also cross at the same point as the main line beams.

To align the receiving optics, a piece of translucent tape is placed at the beam intersection point to scatter the laser light. By tracking the scattered light, the receiving optics are aligned to give a sharp image at the pin-hole aperture in front of the photomultiplier tubes.

4.2 Signal Processing

The sensitivity of the photomultiplier (PM) tubes used in the receiving optics can be varied by varying the applied voltage. Typically, a voltage of about 1000 volts is applied. This voltage can be increased to make the PM tubes more sensitive, as long as the threshold levels on the processors are increased accordingly, since the amount of noise picked up is also increased.

The measured signal is mixed electronically with the signal from a sine wave generator. The frequency of the sine wave (the mixing frequency) is chosen based on the expected flow velocities. The mixing frequency is chosen such that the difference between it and the Bragg shift is about twice the maximum expected Doppler frequency. This allows enough margin for fluctuations about the expected Doppler frequency and still have a remaining nonzero resultant frequency \( f_{\text{res}} = f_{\text{bragg}} - f_{\text{mix}} - f_{\text{dopp}} \). If \( f_{\text{mix}} \) is too high, a biasing results, similar to the fringe biasing caused by stationary fringes. The number of fringes crossed by a particle per second (as seen by the processor) is proportional to \( \Delta t (f_{\text{bragg}} - f_{\text{mix}}) \), where \( \Delta t \) is the time taken by a particle to cross one fringe. Noting that \( \Delta t \) is only determined by the fringe spacing and the flow speed, if \( f_{\text{mix}} \) is increased, the number of fringes crossed by a particle is effectively reduced. This means that signals from particles which cross the fringes at an angle may not have enough fringe crossings (8) to be validated, and hence a bias towards particles moving perpendicularly to
the fringes (higher velocity) results. This gives a higher mean velocity but a lower fluctuation level.

The amplitude of the sine wave must be chosen so that an adequate signal-to-noise ratio is maintained. Typically, a peak-to-peak amplitude of 1 volt is required.

The mixed signal is then fed into single particle burst (Macrodyne) processors. The high pass and low pass filter frequencies are set so that the processor frequency is centered between the two. Usually the high pass filter is set at 0.5 Mhz, which is the lowest non-zero setting, and the low pass filter is set anywhere from 2 to 16 Mhz, depending on the magnitude of the velocity component being measured. The filter bandwidth should be broad enough so that no parts of the fluctuating signal are attenuated.

The gain on the processor is normally set to 10. The output of the processors is displayed on an oscilloscope, and the Doppler signals should read about 1 volt peak-to-peak. The PM tube voltage can be adjusted so that the signal is at the desired level. Signal levels of more than about 1 volt end up being clipped and will therefore not be validated by the processor.

The comparator accuracy for the 5/8 signal validation test can be set between 0 and 10 count variation. The 0 setting is the most accurate, and 11 is off. The processor manufacturer recommends that this level be usually set to 9.

The range on the processor sets the bandwidth of frequencies that the processor can see, according to Table 1. For each range, a particular frequency corresponds to a count, from 0 to 1023. The processor frequency should be matched to a number in the central column in Table 1, and the corresponding range should be set on the Macrodyne processor. An additional check can be made on the range setting by monitoring the analog output from the Macrodyne on a DC voltmeter. The output ranges from 0 to 10 volts, which corresponds to the counts from 0 to 1023. The correct range setting is that which gives about 5 volts on the voltmeter at the operating velocity.
The threshold is then set for the multi-sequence check. The threshold should be set so that the data rate is about half of the data rate at zero threshold. Another check is to block one beam in each color pair. The data rate on the corresponding channel should be zero when one beam is blocked.

To obtain good data rates in air, the flow must be seeded with uniformly sized particles. Smoke, obtained from burning mineral oil or incense, provides particles of approximately 2 µm.

The computer interface should be set up as shown in Fig. 10. First choose the coincidence mode. If shear stresses are to be evaluated, coincidence on all 3 channels must be selected. The coincidence time (5 to 50 µs) sets the window width during which coincidence is defined. This should be set at 5 µs. If only the individual velocities are desired, then the random mode can be selected. The dead time (5 to 50 µs) should then be set so that data from one particle is not recorded twice. A setting of about 25 µs is recommended for low-speed flows.

The number of words that must be multiplexed can be calculated as follows: #words = #inputs + 2. This number should be rounded to the nearest even number. In the present case a setting of six is used. The event mode is set to LDV (digital data only). The LDV-A/D switch enables both digital and analog data to be interfaced simultaneously. The counter clock frequency is set to equal the approximate data rate, and the computer select is set to HP.

5. INHERENT PROBLEMS

Some design problems inherent to 3-component LDV systems are discussed in this section. One main problem with some of the earlier designs which called for splitting a color to create the third beam pair had to do with cross-talk. This is where signals from the two channels bearing the same color could not be adequately separated. In the original design of the present
set-up, the polarization of one of the green beam pairs was rotated relative to the other, so that the receiving optics could distinguish between the two signals. However, using a relatively large angle for the off-axis beams (45°) and two separate collection lenses, cross-talk between the two green channels has been almost eliminated, thus making the polarization rotation dispensable.

Some earlier designs of 3-component LDV systems measured the \( u + w \) velocity component with the off-axis third beam pair. It is shown in Ref. 1 that measuring the \( v + w \) component instead, as done in the present system, reduces the uncertainty in the \( w \) component relative to the uncertainty in \( w \) from these earlier systems.

Another problem has to do with signal coincidence. Details of the probe volumes for the present system are shown in Fig. 3. It is clearly illustrated how the cross-over region between the three sets of beams forms a very small fraction of the overall probe volume. Thus, with heavy seeding (necessary for three-channel work), the electronics may validate data received from different particles which are not necessarily in the cross-over region but are within the coincidence time window set on the interface. This results in a lack of correlation between the measured velocities, and causes the evaluated shear stresses to be inaccurate. Two schemes have been used in an attempt to minimize this problem. First, the coincidence time was made so short that measurements from different particles may be considered instantaneous. The minimum setting of 5 µs available in the present hardware was used; this is equivalent to about half the flight time of a particle passing through the probe volume. The second procedure involved reducing the effective "viewed" probe length and thereby reducing the probability of this "apparent coincidence" (Fig. 5). A collection angle \( \theta \) of 30° and an aperture diameter of 0.5 mm were used to give an effective length of about 1.5 mm.
6. SAMPLE RESULTS

As an initial check on the accuracy of the present system, some preliminary measurements have been made in a vortex/mixing layer interaction, previously investigated using the cross-wire technique (Ref. 6). Since the induced cross-flow angles in this interaction are only $5^\circ -10^\circ$, cross-wire measurements are expected to be accurate to within about 5%. A schematic of the experimental set-up is shown in Figs. 11a and b. LDV measurements of the secondary flow velocities at one streamwise station ($x = 229$ mm) are presented and compared to the cross-wire measurements in Figs. 12a and b. LDV and cross-wire measurements of the turbulence quantities at one spanwise position ($z = 13$ mm) are compared in Fig. 13.

The secondary flow velocities are qualitatively similar, although the LDV measurements indicate a somewhat higher $\bar{w}$. The normal intensity $w'^2$ also seems slightly high. The higher $\bar{w}$ measurements are more likely caused by a slight misalignment of the beams relative to the tunnel axis rather than by remnants of the apparent coincidence problem (discussed above in Section 5), since the latter would not affect the $\bar{w}$ measurements. $v'^2$ seems to agree very well whereas $u'^2$ is a bit low, and since $\bar{u}$ was a bit high, this was probably a result of fringe biasing caused by too high a mixing frequency (as discussed above in Section 4.2). (The mixing frequencies used for these measurements were 37.5 MHz for the u channel, and 38 MHz for the v and $w_v$ channels.) However, the normal stress measurements agree to within 10%, and the shear stresses are consequently affected. $u'v'$ is somewhat low (about 20%) whereas $u'w'$ is low by almost a factor of two. The measurement of $u'w'$ with the present system seems to be very sensitive. This is due to the fact the $w_v * u$ is of the same order as $u * v \cos 45^\circ$, so any small error in the measurement of these velocities or the off-axis angle can result in large errors in $u'w'$. $v'w'$ in vortex affected flows is generally of the same order as $u'w'$ and this seems to be the case with
the present measurements. With a cross-wire, $v'w'$ has to be evaluated from measurements made in four different planes about the probe axis, and hence was not measured here. The comparisons clearly demonstrate the potential of the system in measuring detailed mean flow and turbulence quantities in three-dimensional flows. Work is in progress on optimizing the problems discussed above so that the measurement accuracies may be improved.

6. CONCLUDING REMARKS

A 3-component LDV system, capable of measuring all three components of velocity simultaneously has been developed for use in low-speed three-dimensional flows. All the six components of Reynolds shear stress and higher order products of interest can hence be evaluated. The approach followed was to convert an existing 2-component system by splitting one of the colors to produce the third beam pair. The additional optical hardware required for this process is relatively minor.

For the first time, three-component measurements made with an LDV system have been compared directly with those obtained with the cross-wire technique, in a three-dimensional flow field where both techniques are expected to perform satisfactorily. The preliminary measurements are encouraging and work is in progress on improving the system accuracy.
APPENDIX

SOFTWARE FOR THE HP 9845B DESK-TOP COMPUTER

Complete listings of two programs written in BASIC and ASSEMBLY languages are included in this appendix: "LDV" for data acquisition, some on-line data reduction, and storage of data on floppy disks; and "STAT" for complete off-line data reduction from files written to disk.
REM PROGRAM LDV
!
PROGRAM TO ACQUIRE DATA FROM THREE-COMPONENT LDV SYSTEM
!
The program asks for initialization data and calculates the

calibration constants from them. It reads 3 channels of raw

LV data from the LDV A/D CI and writes them to a disk file

together with the calibration constants. The program can reduce
the data and display real-time histograms of the raw LV data
if desired.

OPTION BASE 1

ICOM 16600
IDELETE ALL
IASSEMBLE Find_vel
IASSEMBLE Data_trans
IASSEMBLE Draw_hist

COM INTEGER Data(3,2000),D1(2000,6),Ns,Nn ! D1 is data buffer
COM REAL Df1,Df2,Df3
COM REAL Theta,Nub,Numix1,Numix2,Numix3
COM REAL Su,Sv,Sww,Suv,Suw,Svw
DIM Date$(100),File$(16),Name$(4),Titl$(80)
REAL Xpos,Ypos,2pos
INTEGER A,B,Rangel,Range2
INTEGER Run,Dn,Nss,N
REAL Phi1,D,F,Phi2
REAL Lam1,Lam2,Lam3,Db,Prwid1,Pr1en1
REAL Re,Ue
REAL Prwid2,Prwid3,Pr1en2,Pr1en3
REAL Nfr1,Nfr2,Nfr3
REAL U,V,W

Ns=2000

DEG
PRINT
PRINT " ** << PROGRAM LDV : 3-COMPONENT VELOCITY DATA >> ** "
PRINT
PRINT "PROGRAM STRUCTURE"
PRINT " 1. INITIALIZE VARIABLES AND CALCULATE PARAMETERS"
PRINT " 2. ACQUIRE DATA FROM A/D"
PRINT " 3. WRITE TO FLOPPY DISC"
PRINT
!
** CHECK HISTOGRAMS **
!
Ans$="N"
INPUT "DO YOU WISH TO LOOK AT HISTOGRAMS ? (Y/N, DEFAULT N)" ,Ans$
IF Ans$="Y" THEN GOSUB Hist
!
** INITIALIZE RUN **
!
PRINT " ** INITIALIZATION ** "
Run=1
PRINT " ENTER RUN PARAMETERS:"
PRINT
INPUT "Enter date and time:" ,Date$
INPUT "Enter 1-Line Name For Profile :" ,Titl$
INPUT "No. of data samples per point (2000 maxm.) :" ,Ns
!
BEAM SPACING J IS FIXED
!
D=.013
!
FOCAL LENGTH F IS FIXED
610 !
620 F=.381
630 !
640 ! WAVELENGTHS OF 3 BEAMS ARE FIXED
650 !
660 Lam1=4.98E-7
670 Lam2=5.145E-7
680 Lam3=5.145E-7
690 !
700 ! REFERENCE VELOCITY SET TO ZERO
710 !
720 Ue=0
730 !
740 ! CALCULATE HALF-ANGLES FROM BEAM SPACINGS AND FOCAL LENGTH
750 !
760 Ph1=ATN(D/2/F)
770 Ph2=Ph1*2
780 !
790 ! CALCULATE FRINGE SPACINGS FROM WAVELENGTHS AND HALF-ANGLES
800 !
810 Dfl=Lam1/2/SINC(Ph1)
820 Df2=Lam2/2/SINC(Ph1)
830 Df3=Lam3/2/SINC(Ph1)
840 !
850 ! BEAM DIAMETER Db IS FIXED
860 !
870 Db=1.2E-3
880 !
890 ! CALCULATE PROBE VOLUME WIDTH AND LENGTH
900 !
910 Prwid1=4*Lam1*F/PI/Db/COSC(Ph1)
920 Prlen1=4*Lam1*F/PI/Db/SINC(Ph1)
930 Prwid2=4*Lam2*F/PI/Db/COSC(Ph1)
940 Prlen2=4*Lam2*F/PI/Db/SINC(Ph1)
950 Prwid3=Prwid2
960 Prlen3=Prlen2
970 Nfr1=Prwid1/Dfl
980 Nfr2=Prwid2/Df2
990 Nfr3=Nfr2
1000 !
1010 ! GET MORE RUN PARAMETERS
1020 !
1030 INPUT "Enter Bragg shift frequency (MHz) ":,Nub
1040 Kill_w=3
1050 INPUT "Is this a two-channel or three-channel run (2/3, default 3)?",Kill
1060 IF Kill_w=3 THEN GOTO 1110
1070 INPUT "Enter mixing frequency (MHz, 2 nos.)",Numix1,Numix2
1080 Numix3=0
1090 Theta=0
1100 GOTO 1130
1110 INPUT "Enter mixing frequency (MHz, 3 nos.)",Numix1,Numix2,Numix3
1120 INPUT "Enter 3rd beam angle (degs) ":,Theta
1130 INPUT "Enter tunnel reference voltage (volts) ",Vref
1140 PRINTER IS 13
1150 !
1160 ! PRINT HEADER
1170 !
1180 PRINT Title$
1190 PRINT "TEST DATE AND TIME ":,Date$
1200 PRINT "BEAM SPACINGS (m)= ",D
1210 PRINT "TOTAL ANGLE BETWEEN BEAMS (Degrs) =",Ph2
1220 PRINT "FRINGE SPACINGS (m) =",Df1,Df2,Df3
1230 PRINT "PROBE WIDTHS (m) =",Prwid1,Prwid2,Prwid3
1240 PRINT "PROBE LENGTHS (m) =",Prlen1,Prlen2,Prlen3
1250 PRINT "NO. OF FRINGES =",Nfr1,Nfr2,Nfr3
1260 PRINT "BRAgg SHIFT FREQUENCY (MHz) =",Nub
1270 PRINT "MIXING FREQUENCY (MHz) =",Numix1,Numix2,Numix3
1280 PRINT "THIRD BEAM SET ANGLE (Degrs) =",Theta.
1290 PRINT "RUN NUMBER = ",Run
1300 PRINT "TUNNEL REFERENCE VOLTAGE (volts) =",Vref
1310 PRINTER IS 16
1320 Ans$="N"
1330 INPUT "DO YOU WISH TO MAKE ANY CHANGES? (Y/N, DEFAULT N) ",Ans$
1340 IF Ans$="Y" THEN GOTO 540
1350 !
1360 ! DATA ACQUISITION
1370 !
1380 PRINT " ** DATA ACQUISITION ** 
1390 REDIM Data(3,Ns)
1400 Ans$="N"
1410 INPUT "DO YOU WISH TO CHANGE FILE NAME? (Y/N, DEFAULT N) ",Ans$
1420 IF Ans$="N" THEN GOTO 1490
1430 INPUT "Enter 4-digit filename: ",Name$
1440 PRINTER IS 0
1450 PRINT "4-digit filename for profile ": ,Name$
1460 INPUT "Disk Number =",Dn
1470 PRINTER IS 16
1480 INPUT "Enter X, Y, AND Z locations: ",Xpos,Ypos,Zpos
1490 !
1500 ! TAKE DATA
1510 !
1520 !
1530 ! PRINTER IS 0
1540 PRINT
1550 PRINT
1560 PRINT
1570 PRINT "POINT NUMBER IN PROFILE: ",Run
1580 PRINT "X,Y,Z = ",Xpos,Ypos,Zpos
1590 PRINTER IS 16
1600 GOSUB Atod
1610 !
1620 ! CALCULATE SAMPLE VELOCITIES
1630 !
1640 Ans$="N"
1650 INPUT "DO YOU WISH TO OBTAIN ESTIMATES OF U AND V? (Y/N, DEFAULT N) ",Ans$
1660 IF Ans$="N" THEN GOTO 1720
1670 RAD
1680 ICALL Find_vel
1690 GOTO 2420
1700 !
1710 ! WRITE CALIBRATION CONSTANTS AND DATA TO DISK
1720 !
1730 GOSUB Dfile
1740 Ans$="Y"
1750 INPUT "DO YOU WISH TO TAKE ANOTHER POINT? (Y/N, DEFAULT Y) ",Ans$
1760 IF Ans$="N" THEN GOTO 1820
1770 Run=Run+1
1780 Ans$="N"
1790 INPUT "DO YOU WISH TO CHANGE ANY PARAMETERS? (Y/N, DEFAULT N) ",Ans$
1800 IF Ans$="Y" THEN GOTO 540
1810 GOTO 1410
1820 END
1830 ! .................................................. ! END OF MAIN PROGRAM LDV ! .................................................. 
1840 ! ..................................................
1850 ! ..................................................
1860 Atod: ! Subroutine for input from the LDV-A/D CI
1870 ! Enter routine with Ns = no. samples
1880 DISP "Press CONT to initiate data acquisition"
1890 PAUSE
1900 DISP "Acquiring Data"
1910 RESET 10
1920 CONTROL MASK 10;1
1930 WRITE IO 10,5;0                      ! start handshake by setting CTL0
1940 Nt=6*Ns
1950 FOR I=1 TO 5
1960 Dummy=READBIN(10)
1970 NEXT I
1980 REDIM D1(Ns,6)
1990 ENTER Ie WFHS Nt NOFORMAT;D1(*) ! fast data acquisition
2000 WRITE IO 10,5;0
2010 PRINT
2020 DISP "Data acquisition complete"
2030 PRINT
2040 ICALL Data_trans; RETURN
2050 ! ** DATA FILE WRITE TO FLOPPY DISK **
2060 PRINT
2070 PRINT "At this point be sure there is a floppy in drive 0 of" 
2080 PRINT "the 9895A with space for a file of 101, 256-byte records."
2090 PRINT
2100 PRINT
2110 PRINT
2120 PRINT "DO YOU WISH TO WRITE THESE DATA TO DISK ? (Y/N, DEFAULT Y)" ,Ans$ 
2130 IF Ans$="N" THEN GOTO 2380
2140 File$=Name$&VAL$(Run)
2150 WRITE IO 10,5;0
2160 MASS STORAGE IS ":H8,0,0" ! set floppy drive (9895A drive 0) as default
2170 CREATE File$,101  
2180 ASSIGN File$ TO #1
2190 PRINT #1; Date$ 
2200 PRINT #1; Title$
2210 PRINT #1; Name$
2220 PRINT #1; Dn
2230 PRINT #1; Hk, Vref, Uv, Df1, Df2, Df3
2240 PRINT #1; Xpos, Ypos, Zpos, Run
2250 MAT PRINT #1; Data
2260 PRINT "**** File write completed ****"
2270 ASSIGN * TO #1            ! close data file
2280 MASS STORAGE IS ":H8,0,1" ! reset program disk as mass storage
2290 GOTO 2390
2300 Run=Run-1
2310 RETURN
2400 ! .................................................. !
2410 Uguess: ! Estimate U-component of velocity
2420 DISP "CALCULATING ESTIMATES OF U AND V VELOCITIES"
2430 !
2440 !
2450 IF Kill_w=3 THEN GOTO 2490
2460 DATA 0,0,0
2470 RESTORE 2460
2480 READ Sw,Sww,Suw
2490 Ubar=Su/Ns
2500 Vbar=Sv/Ns
2510 Wbar=Sw/Ns
2520 Upri2=Suu/Ns-Ubar*Ubar
2530 Vpri2=Svv/Ns-Vbar*Vbar
2540 Wpri2=Sww/Ns-Wbar*Wbar
2550 Uubar=Sw/Ns-Ubar*Wbar
2560 Vvbar=Svw/Ns-Vbar*Wbar
2570 Vwbar=Suv/Ns-Vbar*Wbar
2580 PRINTER IS 0
2590 PRINT "ESTIMATE OF QUANTITIES FROM SAMPLES ":",Ns
2600 PRINT "Ubar =",Ubar
2610 PRINT "Vbar =",Vbar
2620 PRINT "Wbar =",Wbar
2630 PRINT "Upri2 =",Upri2
2640 PRINT "Vpri2 =",Vpri2
2650 PRINT "Wpri2 =",Wpri2
2660 PRINT "Uubar =",Uubar
2670 PRINT "Uvbar =",Uvbar
2680 PRINT "Vvbar =",Vvbar
2690 GOTO 1700
2700 ISOURCE !
2710 ISOURCE ! This subroutine converts raw data counts into instantaneous velocities, then sums several different products of the velocity components. All input and output data is passed through the COMMON storage area. The inputs are the raw data array (Arrayd), the Bragg shift frequency (Nub), the mixing frequencies (Numix1, Numix2, Numix3), the fringe spacings (Df1, Df2, Df3), the crossing angle of the third beam (Theta), and the number of samples in a data point (Ns). The outputs are the summations of various products of the velocity components, including U, V, W, U*U, V*V, W*W, U*V, U*W, and V*W.
2730 ISOURCE !
2740 ISOURCE ! Declare subroutines stored outside of the main program.
2750 ISOURCE ! Declare common variables.
2760 ISOURCE !
2770 ISOURCE Data_par: INT (*) ! Declare common variables.
3010 ISOURCE Hmix1_par: REL
3020 ISOURCE Hmix2_par: REL
3030 ISOURCE Hmix3_par: REL
3040 ISOURCE Su_par: REL
3050 ISOURCE Sv_par: REL
3060 ISOURCE Sw_par: REL
3070 ISOURCE Suu_par: REL
3080 ISOURCE Svv_par: REL
3090 ISOURCE Sww_par: REL
3100 ISOURCE Suv_par: REL
3110 ISOURCE Suw_par: REL
3120 ISOURCE Svw_par: REL
3130 ISOURCE \\
3140 ISOURCE Arrayd: BSS 39 ! Reserve space for data array
3150 ISOURCE Elementd: EQU Arrayd+16 ! descriptor.
3160 ISOURCE Array1: BSS 4096 ! Reserve space for lookup tables used
3170 ISOURCE Array2: BSS 4096 ! for count to velocity conversion.
3180 ISOURCE Array3: BSS 4096
3190 ISOURCE Ns: BSS 1 ! Reserve space for various input
3200 ISOURCE Df1: BSS 4 ! and output variables.
3210 ISOURCE Df2: BSS 4
3220 ISOURCE Df3: BSS 4
3230 ISOURCE Theta: BSS 4
3240 ISOURCE Nub: BSS 4
3250 ISOURCE Numix1: BSS 4
3260 ISOURCE Numix2: BSS 4
3270 ISOURCE Numix3: BSS 4
3280 ISOURCE Su: BSS 4
3290 ISOURCE Sv: BSS 4
3300 ISOURCE Sw: BSS 4
3310 ISOURCE Suu: BSS 4
3320 ISOURCE Svv: BSS 4
3330 ISOURCE Sww: BSS 4
3340 ISOURCE Suv: BSS 4
3350 ISOURCE Suw: BSS 4
3360 ISOURCE Svw: BSS 4
3370 ISOURCE Count: BSS 1 ! Count and I are general purpose index vari-
3380 ISOURCE I: BSS 1 ! ables. Count is usually 0, 1, or 2, to denote
3390 ISOURCE Check: BSS 1 ! whether U, V, or W is being calculated.
3400 ISOURCE Int: BSS 1 ! Int, Address, and Offset are all general
3410 ISOURCE Address: BSS 1 ! purpose storage areas.
3420 ISOURCE Offset: BSS 1
3430 ISOURCE R1: BSS 4 ! R1, R2, and R3 are the count-to-
3440 ISOURCE R2: BSS 4 ! frequency conversion factors.
3450 ISOURCE R3: BSS 4
3460 ISOURCE Xvar: BSS 4 ! Xvar and Yvar are general purpose real
3470 ISOURCE Yvar: BSS 4 ! number storage areas.
3480 ISOURCE U: BSS 4 ! reserve space for instantaneous velocity
3490 ISOURCE V: BSS 4 ! components.
3500 ISOURCE W: BSS 4
3510 ISOURCE Uu: BSS 4
3520 ISOURCE Uv: BSS 4
3530 ISOURCE Cost: BSS 4 ! Cos and Sin are the cosine and sine of Theta.
3540 ISOURCE Sini: BSS 4
3550 ISOURCE Rad: DAT 5.729578E1
3560 ISOURCE Mill: DAT 1.E6
3570 ISOURCE One: DAT 1.
3580 ISOURCE Zero: DAT 0.
3590 ISOURCE !
3600 ISOURCE SUB
ISOURCE Find_vel:
LDA =Ns  ! Get number of samples.
LDB =Np
JSM Get_value
LDA =Array
JSM Get_value
LDA =Data
JSM Get_info
LDA =Df1
JSM Get_value
LDA =Df2
JSM Get_value
LDA =Df3
JSM Get_value
LDA =Theta
JSM Get_value
LDA =NuB
JSM Get_value
LDA =Nub
JSM Get_value
LDA =Numix1
JSM Get_value
LDA =Numix2
JSM Get_value
LDA =Numix3
JSM Get_value

ISOURCE ! The loop headed by Get_freq is repeated three times to get
3920 ISOURCE ! the count-to-frequency conversion factors (which depend on
3930 ISOURCE ! the range) for U, V, and W. Whenever a loop is controlled by
3940 ISOURCE ! the variable "Count", the loop contains operations which are
3950 ISOURCE ! the same for U, V, and W.
3960 ISOURCE !
3970 ISOURCE !
3980 ISOURCE
LDA =0
STA Count
4000 ISOURCE Get_freq:
LDA Count  ! Get the first word of the column of
LDB Ns  ! the data array which contains the
4010 ISOURCE
LDB Ns  ! velocity component for which we want
4020 ISOURCE
LDB Np  ! to get the range.
4030 ISOURCE
LDB Arrayd  ! Mask and rotate to get the four
4040 ISOURCE
LDA =Int
4050 ISOURCE
LDB =Arrayd  ! bits containing the range.
4060 ISOURCE
JSM Get_element
4070 ISOURCE
LDA Int  ! Subtract from 15 to get the
4080 ISOURCE
LDB =15360  ! actual range.
4090 ISOURCE
AND B
4100 ISOURCE
SAR 10
4110 ISOURCE
TCA  ! Use the range to find the power
4120 ISOURCE
LDB =15  ! of two needed for the divisor.
4130 ISOURCE
ADA B
4140 ISOURCE
LDB =1
4150 ISOURCE
SZA Loopend
4160 ISOURCE Loop:
SBL 1  ! Use the range to find the power
4170 ISOURCE
DSZ A  ! of two needed for the divisor.
4180 ISOURCE
JMP Loop
4190 ISOURCE Loopend:
STB Int
4200 ISOURCE
LDA =Int  ! Convert the power of two into a
STA Oper_1  ! real number.
LDA =Yvar
STA Result
JSM Int_to_REL
LDA =3.2E4  ! Divide 3.2E4 by the appropriate
LDB =Xvar  ! power of two, using BCD math.
XFR 4
STB Oper_1
LDA =Yvar
STA Oper_2
LDA Count  ! Decide whether to put the result
LDB =4  ! in R1, R2, or R3, depending on
MPY  ! Count.
ADA =R1
STA Result
LDA =2
LDA =147155B  ! Now, finally, call the utility to
JSM Rel_math  ! perform the division.
ISZ Count
LDA =3  ! Increment and check Count so as
CPA Count  ! to follow the loop three times.
JMP ++2
JMP Get_freq
LDA =Arrayl  ! Zero out the entire count-to-
LDB =768  ! velocity conversion table so that
CLR 16  ! it must be recalculated for each
ADA =16  ! point. (This must be done if the
DSZ B  ! mixing frequencies or ranges are
JMP Continue  ! changed between counts.)
LDA =Su  ! Set initial values of Su, Sv,
LDB =9  ! Sw, Suu, etc. to zero.
CLR 4  !
ADA =4  !
DSZ B  !
JMP Clear  !
LDA =Theta  ! Convert Theta from degrees to
STR Oper_1  ! radians using the Rel_math
LDA =Rad  ! utility.
STR Oper_2
LDA =Xvar
STA Result
LDA =2
LDB =147155B
JSM Rel_math
LDA =Xvar  ! Find the sine and cosine of
STB Oper_1  ! Theta, and store them in the
LDA =Sin  ! locations Sin and Cos, respectively.
The loop defined by Get_int calculates intermediate values used in converting counts to velocities. These values are the same for all samples in a point, so they can be calculated separately. The loop calculates \( (\text{Num}_N - \text{Num}_1) \) and \( (\text{Mill}_N \times \text{Df}_N) \), where \( N \) is 1, 2, and 3.

The loop \( \text{Begin} \) is performed three times, once for \( U \), \( V \), and \( W \). Each time through the data array is read, a count (a raw datum) is taken from it, and is converted into a velocity. The velocity is stored in \( U \), \( V \), or \( W \) depending on whether this is the first, second, or third iteration of the loop. The first time any particular count is encountered, the velocity corresponding to it is calculated using the intermediate values found in Get_int, and the velocity is stored in a table. If that count is found again in the data array, the corresponding velocity is looked up rather than being calculated again.

The loop \( \text{Begin} \) is performed three times, once for \( U \), \( V \), and \( W \). Each time through the data array is read, a count (a raw datum) is taken from it, and is converted into a velocity. The velocity is stored in \( U \), \( V \), or \( W \) depending on whether this is the first, second, or third iteration of the loop. The first time any particular count is encountered, the velocity corresponding to it is calculated using the intermediate values found in Get_int, and the velocity is stored in a table. If that count is found again in the data array, the corresponding velocity is looked up rather than being calculated again.
5410 ISOURCE STA Elementd
5420 ISOURCE LDA =Int       ! Get a raw datum from the data array.
5430 ISOURCE LDB =Arrayd
5440 ISOURCE JSM Get_element
5450 ISOURCE LDA Int
5460 ISOURCE LDB =1023       ! Strip off the first six
5470 ISOURCE AND B           bits of the raw data word.
5480 ISOURCE LDB Count      ! See if Count = Check.
5490 ISOURCE TCB
5500 ISOURCE ADB Check
5510 ISOURCE S2B Straight   ! If true, use the stripped data
5520 ISOURCE TCA            word as an index. If not, use
5530 ISOURCE ADB =1024      ! 1024 minus the data word.
5540 ISOURCE Straight: STA Int  ! Store the count we have gotten.
5550 ISOURCE LDA Count      ! Now, use Count to find out
5560 ISOURCE LDB =4096       ! which lookup table array
5570 ISOURCE MPY            ! we want to use, and use the
5580 ISOURCE LDB Int        ! count we got from the data array
5590 ISOURCE ADB =-1        ! to find exactly where in the table
5600 ISOURCE SBL 2          ! we want to go.
5610 ISOURCE ADB B
5620 ISOURCE ADB =1
5630 ISOURCE ADB =Array1
5640 ISOURCE STA Address
5650 ISOURCE LDA Address,1  ! If that table entry is zero,
5660 ISOURCE S2A Calculate  ! calculate a velocity for it.
5670 ISOURCE JMP Over
5680 ISOURCE Calculate: LDA =Int
5690 ISOURCE STA Oper_1     ! Convert the count to a real number.
5700 ISOURCE LDA =Yvar
5710 ISOURCE STA Result
5720 ISOURCE JSM Int_to_re1
5730 ISOURCE STB Oper_2
5740 ISOURCE LDA =R1        ! Divide the range we found
5750 ISOURCE ADB Offset    ! earlier by the count to get a
5760 ISOURCE STA Oper_1    ! frequency.
5770 ISOURCE LDA =Xvar
5780 ISOURCE STA Result
5790 ISOURCE LDA =2
5800 ISOURCE LDB =147155B
5810 ISOURCE JSM Rel_math
5820 ISOURCE !
5830 ISOURCE LDA =Numix1    ! Find (Nub-NumixN)-FrequencyN.
5840 ISOURCE ADB Offset
5850 ISOURCE STA Oper_1
5860 ISOURCE LDA =Xvar
5870 ISOURCE STAOper_2
5880 ISOURCE LDA =Yvar
5890 ISOURCE STA Result
5900 ISOURCE LDA =2
5910 ISOURCE LDB =146717B
5920 ISOURCE JSM Rel_math
5930 ISOURCE !
5940 ISOURCE LDB Count     ! If we are calculating U, reverse
5950 ISOURCE ADB =-1       ! the sign of (Nub-NumixN)-FrequencyN
5960 ISOURCE SBM *+2       ! so as to reverse the sign of U.
5970 ISOURCE JMP Samesign  ! Leave V and W alone.
5980 ISOURCE LDA =Zero
5990 ISOURCE STA Oper_1
6000 ISOURCE LDA Result
LDA Result

Find Velocity = ((Nub - NumixN) - FrequencyN) * (Mill * DfN)
and store in a place in the lookup table corresponding to
the data count.

LDA Result

Now transfer the velocity from
the lookup table to U, V, or W,
as appropriate.

LDA Address

Have U, V, and W all
been calculated?

Pause:

If not, go back again.

If so, set Count = 0.

Now we convert the W we have obtained (which is measured at
an angle to the V-axis) to the W we want (which should
be measured at an angle of 90 degrees to the V-axis).
Thus find W = (Wv - V * Cos(Theta)) / Sin(Theta).

LDA Yvar

34
STA Oper_2
LDA =W

STA Result
LDA =2

LDB =147155B

JSM Rel_math

Now take running sums of U, V, W, and several products of these velocities. The sums are taken using the utility "Add". The sums are calculated in an unusual sequence in order to reduce the number of steps needed to calculate them. Program steps which produce sums not considered necessary have been included, but these steps are preceded by the "!" comment marker.

LDA =Su
LDB =U
JSM Add

LDA Oper_2
! Find U+U

STA Oper_1
LDA =Uu

STA Result
LDA =2

LDB =147837B

JSM Rel_math

LDA =Suu
! Find Su=Su+(U+U)

LDB Result

JSM Add

LDA =Su
! Find Su=Su+V

LDB =V

JSM Add

LDA =U
! Find U+V

LDB =V

STA Oper_1

JSM Add

LDA =V
! Find V+V

LDB =V

JSM Add

LDA =Suv
! Find Suv=Suv+(U+V)

LDB Result

JSM Add

LDA =V
! Find V+V

LDB =V

JSM Add

LDA =Suv
! Find Suv=Suv+(V+V)

LDB Result

JSM Add

LDA =Suv
! Find Suv=Suv+(V+V)
7210 ISOURCE LDA =Sw ! Find Sw=Sw+W
7220 ISOURCE LDB =W
7230 ISOURCE JSM Add
7240 ISOURCE !
7250 ISOURCE LDA =U ! Find U*W
7260 ISOURCE STA Oper_1
7270 ISOURCE LDA=Xvar
7280 ISOURCE STA Result
7290 ISOURCE LDA =2
7300 ISOURCE LDB =1470378
7310 ISOURCE JSM Rel_math
7320 ISOURCE !
7330 ISOURCE LDA =S uw ! Find Suw=Suw+(U*W)
7340 ISOURCE LDB Result
7350 ISOURCE JSM Add
7360 ISOURCE !
7370 ISOURCE LDA =W ! Find V*W
7380 ISOURCE STA Oper_1
7390 ISOURCE LDA =V
7400 ISOURCE STA Oper_2
7410 ISOURCE LDA=Xvar
7420 ISOURCE STA Result
7430 ISOURCE LDA =2
7440 ISOURCE LDB =1470378
7450 ISOURCE JSM Rel_math
7460 ISOURCE !
7470 ISOURCE LDA =Suw ! Find Suw=Suw+(V*W)
7480 ISOURCE LDB Result
7490 ISOURCE JSM Add
7500 ISOURCE !
7510 ISOURCE LDA =W ! Find W*W
7520 ISOURCE STA Oper_1
7530 ISOURCE STA Oper_2
7540 ISOURCE LDA=Xvar
7550 ISOURCE STA Result
7560 ISOURCE LDA =2
7570 ISOURCE LDB =1470378
7580 ISOURCE JSM Rel_math
7590 ISOURCE !
7600 ISOURCE LDA =Suw ! Find Suw=Suw+(W*W)
7610 ISOURCE LDB Result
7620 ISOURCE JSM Add
7630 ISOURCE !
7640 ISOURCE DSZ I ! Continue to calculate running sums until out of samples.
7650 ISOURCE JMP Begin ! sums until out of samples.
7660 ISOURCE !
7670 ISOURCE ! Place the finished sums in the COMMON region so that
7680 ISOURCE ! the BASIC program has access to them, and then return to
7690 ISOURCE ! the BASIC program.
7700 ISOURCE !
7710 ISOURCE LDA =Su
7720 ISOURCE LDB =Su_par
7730 ISOURCE JSM Put_value
7740 ISOURCE LDA =Sw
7750 ISOURCE LDB =Sw_par
7760 ISOURCE JSM Put_value
7770 ISOURCE LDA =Sw
7780 ISOURCE LDB =Sw_par
7790 ISOURCE JSM Put_value
7800 ISOURCE LDA =Suw
7810 ISOURCE LDB =Suu_par
7820 ISOURCE JSM Put_value
7830 ISOURCE LDA =Suv
7840 ISOURCE LDB =Suv_par
7850 ISOURCE JSM Put_value
7860 ISOURCE LDA =Suv
7870 ISOURCE LDB =Suv_par
7880 ISOURCE JSM Put_value
7890 ISOURCE LDA =Sww
7890 ISOURCE LDB =Sww_par
7890 ISOURCE JSM Put_value
7900 ISOURCE LDA =Suv
7910 ISOURCE LDB =Suv_par
7920 ISOURCE JSM Put_value
7930 ISOURCE LDA =Suv
7940 ISOURCE LDB =Suv_par
7950 ISOURCE JSM Put_value
7960 ISOURCE LDA =Suv
7970 ISOURCE LDB =Suv_par
7980 ISOURCE JSM Put_value
7990 ISOURCE ! The utility "Add" is used to add up the running sums.
8000 ISOURCE !
8020 ISOURCE Add: ISZ Utlcount
8030 ISOURCE STA Oper_1
8040 ISOURCE STB Oper_2
8050 ISOURCE STA Result
8060 ISOURCE LDA =2
8070 ISOURCE LDB =146721B
8080 ISOURCE JSM Rel_math
8090 ISOURCE DSZ Utlcount
8100 ISOURCE RET 1
8110 ISOURCE !
8130 ISOURCE LIT 200
8140 ISOURCE END Find_vel
8150 ISOURCE !
8160 ISOURCE !
8170 ISOURCE ! Subroutine to transfer raw data from input array
8190 ISOURCE ! to storage array.
8200 ISOURCE !
8210 ISOURCE EXT Get_info ! Declare subroutines stored
8220 ISOURCE EXT Get_value ! outside of the main program.
8230 ISOURCE EXT Get_element
8240 ISOURCE EXT Put_element
8250 ISOURCE !
8260 ISOURCE COM
8270 ISOURCE Data_par: INT (*) ! Declare common variables.
8280 ISOURCE Di_par: INT (*)
8290 ISOURCE Ns_par: INT
8300 ISOURCE !
8310 ISOURCE Arrayd: BSS 39
8320 ISOURCE Elementd: EQU Arrayd+16
8330 ISOURCE Array1: BSS 39
8340 ISOURCE Element1: EQU Array1+16
8350 ISOURCE Ns: BSS 1
8360 ISOURCE Count1: BSS 1
8370 ISOURCE Count2: BSS 1
8380 ISOURCE It: BSS 1
8390 ISOURCE Int: BSS 1
8400 ISOURCE !
8410 ISOURCE SUB
8420 ISOURCE Data_trans: LDA =Arrayd ! Get parameters of data storage array.
8430 ISOURCE LDB =Data_par
8440 ISOURCE JSM Get_info
8450 ISOURCE LDA =Array1 ! Get parameters of data input array.
8460 ISOURCE LDB =Di_par
8470 ISOURCE JSM Get_info
8480 ISOURCE LDA =Ns ! Get number of samples.
8490 ISOURCE LDB =Ns_par
8500 ISOURCE JSM Get_value
8510 ISOURCE !
8520 ISOURCE LDA =0
8530 ISOURCE STA I
8540 ISOURCE Start1: LDA =0
8550 ISOURCE STA Count1
8560 ISOURCE LDA =2
8570 ISOURCE STA Count2
8580 ISOURCE !
8590 ISOURCE Start2: LDA I
8600 ISOURCE LDB =6
8610 ISOURCE MPY
8620 ISOURCE ADA Count2
8630 ISOURCE STA Element1
8640 ISOURCE LDA =Int
8650 ISOURCE LDB =Array1
8660 ISOURCE JSM Get_element
8670 ISOURCE !
8680 ISOURCE LDA Count1
8690 ISOURCE LDB Ns
8700 ISOURCE MPY
8710 ISOURCE ADA I
8720 ISOURCE STA Elementd
8730 ISOURCE LDA =Int
8740 ISOURCE LDB =Arrayd
8750 ISOURCE JSM Put_element
8760 ISOURCE ISZ Count1
8770 ISOURCE ISZ Count2
8780 ISOURCE LDA =-3
8790 ISOURCE ADA Count1
8800 ISOURCE SZA =2
8810 ISOURCE JMP Start2
8820 ISOURCE !
8830 ISOURCE ISZ I
8840 ISOURCE LDA I
8850 ISOURCE TCA
8860 ISOURCE ADA Ns
8870 ISOURCE SZA ++2
8880 ISOURCE JMP Start1
8890 ISOURCE RET 1
8900 ISOURCE END Data_trans
8910 !
8920 !*****************************************************************************************
8930 ! ** HISTOGRAM SUBPROGRAM **
8940 !*****************************************************************************************
8950 !
8960 Hist: ! Subroutine to produce online
8970 ! histograms of raw LDV data.
8980 ON KEY #0 GOTO 9270
8990 Nh=0
9000 Ns_temp=Ns
9010 !
9020 INPUT "No. of data samples per point (2000 maxm.) :", Ns
9030 !
9040 REDIM Data(3,Ns)
9050 REDIM Df(Ns,6)
9060 DISP "Press CONT to initiate data acquisition, press K0 to return to main program."
9070 PAUSE
9080 GCLEAR
9090 GRAPHICS
9100 !
9110 RESET 10
9120 CONTROL MASK 10;1
9130 WRITE 10 10,5;0
9140 WRITE 10 10,5;1 ! start handshake by setting CTL0
9150 Nt=6*Ns
9160 FOR I=1 TO 5
9170 Dummy=READBIN(10)
9180 NEXT I
9190 !
9200 ENTER 10 WFHS Nt NOFORMAT;D1(*) !fast data acquisition
9210 WRITE 10 10 10,5;0
9220 ICALL Data_trans
9230 ICALL Draw_hist
9240 IF Ns=2000 THEN GOTO 9270
9250 Nn=1
9260 GOTO 9110
9270 IF Ns=2000 THEN DUMP GRAPHICS
9280 WRITE 10 10,5;0
9290 Ns=Ns_temp
9300 REDIM Data(3,Ns)
9310 REDIM Df(Ns,6)
9320 EXIT GRAPHICS
9330 RETURN
9340 !
9350 ISOURCE
9360 ISOURCE !
9370 ISOURCE ! The subroutine works by first going to the samples-per-count
9380 ISOURCE ! tables and using them to draw the old histograms in black
9390 ISOURCE ! (Bit=0) to erase them. Then it calculates new samples-per-
9400 ISOURCE ! count tables from the data acquired from the LDV, and uses
9410 ISOURCE ! the new tables to draw histograms in white (Bit=1). Then the
9420 ISOURCE ! subroutine returns to the main program. The very first time
9430 ISOURCE ! (determined by Ntimes(=Nn)) the subroutine is called it doesn’t
9440 ISOURCE ! erase the old histograms because there aren’t any.
9450 ISOURCE !
9460 ISOURCE EXT Get_value ! Declare subroutines stored
9470 ISOURCE EXT Get_info ! outside of the main program.
9480 ISOURCE EXT Get_element
9490 ISOURCE !
9500 ISOURCE !
9510 ISOURCE Data_par: INT (*) ! Declare common variables.
9520 ISOURCE Dl_par: INT (*)
9530 ISOURCE Ns_par: INT
9540 ISOURCE Ntimes_par: INT
9550 ISOURCE !
9560 ISOURCE Array: BSS 39 ! Reserve space for data array
9570 ISOURCE Element: EQU Array+16 ! descriptor.
9580 ISOURCE Array1: BSS 1024 ! Reserve space for tables which hold
9590 ISOURCE Array2: BSS 1024 ! the number of samples per count.
9600 ISOURCE Array3: BSS 1024

39
ISOURCE Ns: BSS 1
ISOURCE Count: BSS 1 ! Count and I are general purpose index vari-
ISOURCE I: BSS 1 ! ables. Count is usually 0, 1, or 2, to denote
ISOURCE Check: BSS 1 ! whether U, V, or W is being calculated.
ISOURCE Int: BSS 1 ! Int, Address, and Offset are all general
ISOURCE Address: BSS 1 ! purpose storage areas.
ISOURCE X_coord: BSS 1
ISOURCE Y_coord: BSS 1
ISOURCE Bit: BSS 1
ISOURCE Ntimes: BSS 1
ISOURCE !
ISOURCE LIT 30
ISOURCE !
ISOURCE SUB
ISOURCE Draw_hist: LDA =Ns ! Get number of samples.
ISOURCE LDB =Ns_par
ISOURCE JSM Get_value
ISOURCE LDA =Arrayd ! Get parameters of data array.
ISOURCE LDB =Data_par
ISOURCE JSM Get_info
ISOURCE LDA =Ntimes ! Get Ntimes, which tells if this is the first time Draw_hist is being called.
ISOURCE LDB =Ntime_par
ISOURCE JMP Acquire
ISOURCE !
ISOURCE Do_graph: LDA =0 ! If this is the first time Draw_hist is being called, then jump to the data acquisition section. If not then write over the old histogram with black to erase it.
ISOURCE STR Bit
ISOURCE LDA Ntimes
ISOURCE JMP Acquire ! Write over the old histogram with black to erase it.
ISOURCE !
ISOURCE Make_hist: LDA Count ! Go to the end of the appropriate samples-per-count table.
ISOURCE ADA =1
ISOURCE LDB =1024
ISOURCE STB I
ISOURCE MPy
ISOURCE ADA =Arrayl
ISOURCE ADA =-1
ISOURCE Address
ISOURCE Make_rod: LDB Address, I ! Add together each adjacent pair of entries in the table.
ISOURCE DSZ Address
ISOURCE ADB Address, I
ISOURCE DSZ Address
ISOURCE DSZ I
ISOURCE RZB ++2 ! If the result is zero, go to the next pair of entries. If not, make sure the sum is <150 and then draw a column with height equal to the sum.
ISOURCE JMP Skip_bits
ISOURCE STB Int
ISOURCE ADA =-150
ISOURCE SBM ++3
ISOURCE ADA =150
ISOURCE STB Int
ISOURCE !
ISOURCE Acquire
ISOURCE LDA I ! Calculate the X-coordinate of the column.
ISOURCE SAR 1
ISOURCE ADA =20
ISOURCE STB X_coord
ISOURCE !
ISOURCE Make_bit: LDA =150 ! Calculate the Y-coordinate of the top of the column.
ISOURCE LDB Count
10210  ISOURCE  ADB =1
10220  ISOURCE  MPY
10230  ISOURCE  LDB Int
10240  ISOURCE  TCB
10250  ISOURCE  ADA B
10260  ISOURCE  STA Y_coord
10270  ISOURCE  !
10280  ISOURCE  LDA =13  ! Prepare the graphics screen for
10290  ISOURCE  STA Pa  input.
10300  ISOURCE  LDA =51B
10310  ISOURCE  SFC * STA R5
10320  ISOURCE  LDA Y_coord  ! Calculate word location.
10330  ISOURCE  LDB =36
10340  ISOURCE  MPY
10350  ISOURCE  SFC *
10360  ISOURCE  STA R4  ! Input word location.
10370  ISOURCE  STA R7
10390  ISOURCE  ADA B
10380  ISOURCE  CMA
10400  ISOURCE  SBL
10410  ISOURCE  STA R4  ! Input bit location.
10420  ISOURCE  STA R7
10430  ISOURCE  !
10440  ISOURCE  LDB X_coord  ! Calculate bit location.
10450  ISOURCE  AND =1?B
10450  ISOURCE  LDB Bit
10460  ISOURCE  SBL 15
10460  ISOURCE  IOR B
10470  ISOURCE  SFC *
10480  ISOURCE  STA R4  ! Input bit location.
10500  ISOURCE  STA R7
10510  ISOURCE  DSZ Int
10520  ISOURCE  DSZ Skip_bits:  DSZ 1
10530  ISOURCE  ISZ Count  ! Make histograms for all three
10540  ISOURCE  Make_hist
10550  ISOURCE  ISZ I  ! channels.
10560  ISOURCE  JMP Make_rod
10570  ISOURCE  ISZ Count
10580  ISOURCE  LDA =-3  ! If Bit=1 then the new histograms
10590  ISOURCE  ADA Count  have been drawn. Go back to the
10600  ISOURCE  AD 2 ++
10610  ISOURCE  JMP Stop  main program. If not, continue.
10620  ISOURCE  Make_hist
10630  ISOURCE  !
10640  ISOURCE  Acquire:  LDA Bit  ! Zero out the tables which hold the
10650  ISOURCE  SZA ++2  number of samples per count.
10660  ISOURCE  JMP Stop
10670  ISOURCE  !
10680  ISOURCE  LDA =Array1  ! Prepare to write new samples-per-
10690  ISOURCE  CLR 16  ! count tables.
10700  ISOURCE  CLR 16
10710  ISOURCE  ADA =16  !
10720  ISOURCE  DSZ B
10730  ISOURCE  JMP Clear
10740  ISOURCE  !
10750  ISOURCE  LDA =0
10760  ISOURCE  STA Count
10770  ISOURCE  STA Check
10780  ISOURCE  LDA Ns
10790  ISOURCE  STA I
10800  ISOURCE  !

41
Figure out which element of the data array we want to pick up.

Get a raw datum from the data array.

Strip off the first six bits of the raw data word.

See if Count = Check.

If true, use the modified data word as an index. If not, use 11324 minus the data word.

Store the count we have gotten.

Increment the appropriate place in the samples-per-count table by one.

If U, V, and W all been done?

If not, go back again.

If so, set Count=0.

Continue to fill samples-per-count tables until out of samples.

Now set Bit=1 and go back and draw the new histograms in white.
REM PROGRAM STAT

PROGRAM TO REDUCE RAW DATA FROM THREE-COMPONENT LDV SYSTEM

Input is 3 channels of raw LV data and calibration constants

Output is the three components of mean velocity, all the components of Reynolds stress and some selected third-order products. Can filter out counts which are excessively far from the mean. Can display histograms of raw and filtered data for all 3 channels. Plots data at end and displays reduced data for entire run. Can write a summary file containing all reduced data and calibration constants.

OPTION BASE 1

ICOM 10000

IDElete ALL

IASSEMBLE Find vel

COM INTEGER Data(3,2000),Ns,Ngs,Countbin(3,1024)

COM INTEGER Range1,RanGe2,Range3

COM REAL Sdev,Df1,Df2,Df3

COM REAL Theta,Nub,Numix1,Numix2,Numix3

COM REAL Su,Su,Su,Sv,Sv,Sv,Sw,Sw,Sw

COM REAL Su,Sv,Sv,Sv,Su,Su,Su,Su,Su

DIM L1(3),R1(3),Li(3),Ui(3),Scale(3)

DIM Date(80),File(6),Name(4),Titl(80)

REAL Xpos,Ypos,Zpos

REAL Point(20,15)

REAL Re,Ue

DATA 13,13,13,13

READ Max_y,Min_y,Filter,Sdev

DATA 75.64,45.64,15.64,99.,69.,39.

MAT READ L1,U1

DATA 5,46,87,41,82,123

MAT READ L1,R1

DATA -.8,2.4,.803,.035,-.0025,.01

READ Li(1),Ui(1),Li(2),Ui(2),Li(3),Ui(3)

DATA .4,.005,.0025

MAT READ Scale

PRINT "** << PROGRAM STAT : 3-COMPONENT VELOCITY DATA >> ** "

PRINT "PROGRAM STRUCTURE"

PRINT "1. Read raw data from floppy disc"

PRINT "2. Convert to velocity"

PRINT "3. Calculate statistics"

PRINT "4. Print results"

PRINT "5. Write to disc file"

PRINT ** Read raw data from floppy disc **

INPUT "Enter parent filename (or E to exit program) ":,Name$

IF Name$="E" THEN GOTO 178

INPUT "Enter no. of first and last data files ",File1,Nf

K11_w=3

INPUT "Is this a two-channel or three-channel run (2/3, Default 3) ?",Kill

Hist$="Y"

43
610 INPUT "Do you wish to see histograms of the data points (Y/N, Default Y)?", Hist$
620 Filter$="N"
630 INPUT "Do you wish to filter the output (Y/N, Default N)?", Filter$
640 IF Filter$="N" THEN GOTO 660
650 INPUT "Enter standard deviation to throw out", Sdev
660 DISP "READING RAW DATA FROM DISK"
670 MASS STORAGE IS ":HS,O,e"
680 PRINTER IS 0
690 Filno=File1
700 IF Filno>Nf THEN GOTO 1740
710 Files=Name$&VAL$(Filno)
720 ASSIGN Files TO #1
730 READ #1;Date$
740 READ #1;Tit1$
750 READ #1;Name$
760 READ #1;Dn
770 READ #1;Nub, Numix1, Numix2, Numix3, Theta, Run
780 READ #1;Vref, Ue, Df1, Df2, Df3
790 READ #1;Xpos, Ypos, Zpos, Ns
800 MAT READ #1;Data
810 PRINT Tit1$
820 PRINT "DATE AND TIME OF TEST ":, Date$
830 PRINT "FILE NAME ON DISK ":, Name$
840 PRINT "DISK NUMBER ":, Dn
850 PRINT "RUN NO. <POINT NO. IN PROFILE > ":, Run
860 PRINT "X, Y, Z = ", Xpos, Ypos, Zpos
870 !
880 ! CALCULATE VELOCITIES
890 DISP "CALCULATING VELOCITIES AND TAKING RUNNING SUMS"
900 !
910 ICALL Find_vel
920 !
930 ! CALCULATE STATISTICS
940 !
950 DISP "CALCULATING STATISTICS"
960 !
970 IF Kill_w=3 THEN GOTO 1030
980 DATA 0,0,0,0,0,0,0,0
990 !
1000 RESTORE 990
1010 READ Su, Sw, Suw, Suv, Suvw, Suw, Suw
1020 !
1030 Ubar=Su/Ngs
1040 Vbar=Sw/Ngs
1050 Wbar=Su/Ngs
1060 Upri2=Sw/Ngs-Ubar*Ubar
1070 Ypri2=Suv/Ngs-Vbar*Vbar
1080 Wpri2=Sw/Ngs-Wbar*Wbar
1090 Uvbar=Suv/Ngs-Ubar*Vbar
1100 Uvbar=Sw/Ngs-Ubar*Wbar
1110 Yvbar=Sw/Ngs-Vbar*Wbar
1120 Uvbar=Suv/Ngs-2*Ubar*Sw/Ngs-Vbar*Suw/Ngs+2*Vbar*Ubar*Ubar
1130 Uvbar=Suv/Ngs-2*Vbar*Suw/Ngs-Ubar*Suv/Ngs+2*Ubar*Vbar*Vbar
1140 Uvbar=Suv/Ngs-2*Ubar*Suv/Ngs-Wbar*Suv/Ngs+2*Wbar*Ubar*Ubar
1150 Uvbar=Suv/Ngs-2*Wbar*Suv/Ngs-Ubar*Suv/Ngs+2*Wbar*Wbar*Wbar
1160 Uvbar=Suv/Ngs-Wbar*Suv/Ngs-Vbar*Suv/Ngs-2*Ubar*Vbar*Wbar
1170 !
1180 ! STORE RESULTS IN ARRAY FOR PLOTTING
1190 !
1200 Fl=Filno=File1+1
1210  Point(CF1,1)=Ypos
1220  Point(CF1,2)=Ubar
1230  Point(CF1,3)=Vbar
1240  Point(CF1,4)=Wbar
1250  Point(CF1,5)=Upri2
1260  Point(CF1,6)=Ypri2
1270  Point(CF1,7)=Wpri2
1280  Point(CF1,8)=Uvbar
1290  Point(CF1,9)=Uwbar
1300  Point(CF1,10)=Vvbar
1310  Point(CF1,11)=Uuvbar
1320  Point(CF1,12)=Uvvbar
1330  Point(CF1,13)=Uuwbar
1340  Point(CF1,14)=Uwwbar
1350  Point(CF1,15)=Uvwbar
1360  IF Ypos<Max_y THEN GOTO 1390
1370  Max_y=Ypos
1380  Min_u=Ubar
1390  IF Ypos>Min_y THEN GOTO 1450
1400  Min_y=Ypos
1410  Max_u=Ubar
1420  ! PRINT RESULTS
1430  !
1450  PRINT "Bragg shift frequency = ",Nub
1460  PRINT "Mixing frequencies=",Numix1,Numix2,Numix3
1470  PRINT "Third beam angle = ",Theta
1480  PRINT "Run number = ",Run
1490  PRINT "Tunnel reference voltage (volts) = ",Vref
1500  PRINT "Fringe spacings = ",Df1,Df2,Df3
1510  PRINT "No. of samples = ",Nc
1520  PRINT "Filter [no. of std. deviations] = ",Sdev
1530  PRINT "Adjusted no. of samples = ",Ncs
1540  PRINT "Ubar = ",PROUNDCUbar,-4)
1550  PRINT "Vbar = ",PROUNDCVbar,-4)
1560  PRINT "Wbar = ",PROUNDCWbar,-4)
1570  PRINT "Upri 2 = ",PROUNDCUpri2,-4)
1580  PRINT "Ypri 2 = ",PROUNDCYpri2,-4)
1590  PRINT "Wpri 2 = ",PROUNDCWpri2,-4)
1600  PRINT "Uvbar = ",PROUNDCUvbar,-4)
1610  PRINT "Uwbar = ",PROUNDCUwbar,-4)
1620  PRINT "Uvbar = ",PROUNDCUvbar,-4)
1630  PRINT "Uuvbar = ",PROUNDCUuvbar,-4)
1640  PRINT "Uwwbar = ",PROUNDCUwwbar,-4)
1650  PRINT "Uvwbar = ",PROUNDCUvwbar,-4)
1660  PRINT "Uuvbar = ",PROUNDCUuvbar,-4)
1670  PRINT "Uwwbar = ",PROUNDCUwwbar,-4)
1680  PRINT "Uvbar = ",PROUNDCUvbar,-4)
1690  PRINT "Uwwbar = ",PROUNDCUwwbar,-4)
1700  PRINT
1710  IF Hist$="Y" THEN GOSUB Histogram
1720  Filno=Filno+1
1730  GOTO 700
1740  Ue=Max_u-Min_u
1750  Plotr$="Y"
1760  INPUT " Do you wish to see the turbulence quantities (Y/N, Default Y) ?",Plotr$  
1770  IF Plotr$="Y" THEN GOSUB Plot
1780  MASS STORAGE IS ":H8,0,1"
1790  PRINTER IS 16
1800 END
1810 !
1820 ! ******************PLOT DATA**************************
1830 !
1840 Plot; PRINTER IS 16
1850 PRINT "Estimate of normalizing velocity is :";Ue
1860 INPUT "Enter correct normalizing velocity if different. ",Ue
1870 Graph$="Y"
1880 INPUT "Do you wish to graph the turbulence quantities (Y/N, Default Y) ?", Graph$
1890 PRINTER IS 0
1900 PRINT "Normalizing velocity is ";Ue
1910 !
1920 FOR I=1 TO 4
1930 DATA 1,2,2,3
1940 READ Exp
1950 Kn=3
1960 IF I=4 THEN Kn=5
1970 FOR J=1 TO Nf-File1+1
1980 FOR K=1 TO Kn
1990 Index=(I-1)*3+K+1
2000 Point(J,Index)=Point(J,Index)/Ue^Exp
2010 IF Index>10 THEN Point(J,Index)=Point(J,Index)*1000
2020 NEXT K
2030 NEXT J
2040 NEXT I
2050 IF Graph$="N" THEN GOTO 2860
2060 !
2070 PLOTTER IS 13,"GRAPHICS"
2080 GRAPHICS
2090 DEG
2100 !
2110 Ymax=Point(1,1)
2120 Ymin=Ymax
2130 FOR J=1 TO Nf-File1+1
2140 IF Point(J,1)>Ymax THEN Ymax=Point(J,1)
2150 IF Point(J,1)<Ymin THEN Ymin=Point(J,1)
2160 NEXT J
2170 !
2180 FOR I=1 TO 3
2190 LIMIT 0,184.47,0,149.8
2200 LOCATE Li(I),Ri(I),39,99
2210 CLIP Li(I),Ri(I),39,99
2220 !
2230 Xmax=Point(1,(I-1)*3+2)
2240 Xmin=Xmax
2250 FOR J=1 TO Nf-File1+1
2260 FOR K=1 TO 3
2270 IF Point(J,(I-1)*3+K+1)>Xmax THEN Xmax=Point(J,(I-1)*3+K+1)
2280 IF Point(J,(I-1)*3+K+1)<Xmin THEN Xmin=Point(J,(I-1)*3+K+1)
2290 NEXT K
2300 NEXT J
2310 !
2320 SCALE Li(I),Ulim(I),Ymin,Ymax
2330 FRAME
2340 AXES Scale(I),.1,0,0
2350 UNCLIP
2360 !
2370 LORG 8
2380 CSIZE 2.5
2390 Ypos=Ymin
2400 MOVE Li(I)+Scale(I)/5,Ypos
! WRITE SUMMARY DATA FILE

INPUT "Do you wish to write a Summary Data File (Y/N Default N)?", Sum$

IF Sum$="N" THEN GOTO 3210

File$=Name$

DISP "File ";File$;" being written to disk"

MASS STORAGE IS ":H8,0,0"

CREATE File$,40

ASSIGN File$ TO #1

PRINT #1;Date$

PRINT #1;Tit1$

PRINT #1;Name$

PRINT #1;In

PRINT #1;Num,Nnumx1,Nnumx2,Nnumx3,Theta,Run

PRINT #1;Vref,Ue,DF1,DF2,DF3

PRINT #1;Nn,File1,Nf

FOR I=1 TO Nf-Nfile+1

FOR J=1 TO 15

PRINT #1;Point(I,J)

NEXT J

NEXT I

RETURN

! *************** DRAW HISTOGRAMS ****************************

Histogram: PLOTTER IS 13,"GRAPHICS"

GRAPHICS

K=1

FOR I=1 TO 3

Cmax=0

FOR J=1 TO 1023 STEP 2

Countbin(I,J)=Countbin(I,J)+Countbin(I,J+1)

IF Countbin(I,J)>Cmax THEN Cmax=Countbin(I,J)

NEXT J

IF (Cmax=0) OR (Cmax=2000) THEN GOTO 3660

LIMIT 0,184.47,0,149.8

LOCATE 7.05,119.63,L11(K),U11(K)

CLIP 7.05,119.63,L11(K),U11(K)

K=K+1

SCALE 1,1024,0,Cmax

FRAME

UNCLIP

LINE TYPE 1

CSIZE 2.8

LRG 8

FOR Y=1 TO 4

Ypos=Y*Cmax/4

FOR J=1 TO 15

PLOT J,Ypos,-2

PLOT -J,Ypos,-1

MOVE 3,Ypos

LABEL PROUND(Ypos,0)

NEXT Y

Ypos=Cmax/20

FOR X=1 TO 8

Xpos=X*128

PLOT Xpos,Ypos,-2

PLOT Xpos,Ypos,-2
3570 PLOT Xpos,-Ypos,-1
3580 MOVE Xpos,-Ypos+1.3
3590 LABEL Xpos
3600 NEXT X
3610 FOR J=1 TO 1023 STEP 2
3620 IF Countbin(I,J)=0 THEN GOTO 3650
3630 MOVE J,0
3640 DRAW J,Countbin(I,J)
3650 NEXT J
3660 NEXT I
3670 MOVE 512,-Ypos*5
3680 LORG 6
3690 LABEL "PLOT OF COUNT VS NUMBER OF SAMPLES PER COUNT"
3700 DUMP GRAPHICS -Ypos*10
3710 RETURN
3720 !
3730 !
3740 ISOURCE
3750 ISOURCE !
3760 ISOURCE ! This subroutine converts raw data counts to instantaneous
3770 ISOURCE ! velocities, then sums several different products of the
3780 ISOURCE ! velocity components. All input and output data is passed
3790 ISOURCE ! through the COMMON storage area. The inputs are the raw
3800 ISOURCE ! data array (Array), the Bragg shift frequency (Nub), the
3810 ISOURCE ! mixing frequencies (Numix1, Numix2, Numix3), the fringe
3820 ISOURCE ! spacings (Df1, Df2, Df3), the crossing angle of the third
3830 ISOURCE ! beam (Theta), and the number of samples in a data point (Ns).
3840 ISOURCE ! The outputs are the summations of various products of the
3850 ISOURCE ! velocity components, including U, V, W, U*U, V*V, W*W,
3870 ISOURCE !
3880 ISOURCE EXT Get_value ! Declare subroutines stored
3890 ISOURCE EXT Get_info ! outside of the main program.
3900 ISOURCE EXT Get_element
3910 ISOURCE EXT Put_element
3920 ISOURCE EXT Int_to_rel
3930 ISOURCE EXT Rel_to_int
3940 ISOURCE EXT Rel_math
3950 ISOURCE EXT Put_value
3960 ISOURCE !
3970 ISOURCE COM
3980 ISOURCE Data_par: INT (*) ! Declare common variables.
3990 ISOURCE Ns_par: INT
4000 ISOURCE Ngs_par: INT
4010 ISOURCE Cbin_par: INT (*)
4020 ISOURCE Ran1_par: INT
4030 ISOURCE Ran2_par: INT
4040 ISOURCE Ran3_par: INT
4050 ISOURCE Sdev_par: REL
4060 ISOURCE Df1_par: REL
4070 ISOURCE Df2_par: REL
4080 ISOURCE Df3_par: REL
4090 ISOURCE Theta_par: REL
4100 ISOURCE Nub_par: REL
4110 ISOURCE Nmix1_par: REL
4120 ISOURCE Nmix2_par: REL
4130 ISOURCE Nmix3_par: REL
4140 ISOURCE Su_par: REL
4150 ISOURCE Su_par: REL
4160 ISOURCE Su_par: REL
Reserve space for data array descriptor.

Reserve space for lookup tables for count to velocity conversion.

Reserve space for table of counts used to generate histograms.

Reserve space for various input and output variables.

Count and I are general purpose index variables. Count is usually 0, 1, or 2 to denote whether U, V, or W is being calculated.

Int, Address, and Offset are all general purpose storage areas.

R1, R2, and R3 are the count-to-frequency conversion factors.

Xvar and Yvar are general purpose real number storage areas.

U, V, and W are the instantaneous velocity components.
4770 ISOURCE W:          BSS 4
4780 ISOURCE Uu:         BSS 4
4790 ISOURCE Vu:         BSS 4
4800 ISOURCE Cos:        BSS 4 ! Cos and Sin are the cos and sin of Theta.
4810 ISOURCE Sin:        BSS 4
4820 ISOURCE Max_u:      BSS 1
4830 ISOURCE Max_v:      BSS 1
4840 ISOURCE Max_w:      BSS 1
4850 ISOURCE Avg_u:      BSS 4
4860 ISOURCE Avg_v:      BSS 4
4870 ISOURCE Avg_w:      BSS 4
4880 ISOURCE Dev_u:      BSS 4
4890 ISOURCE Dev_v:      BSS 4
4900 ISOURCE Dev_w:      BSS 4
4910 ISOURCE Ngsi:       BSS 1
4920 ISOURCE Sdev:       BSS 4
4930 ISOURCE Rad:        DAT 5.729578E1
4940 ISOURCE Mill:       DAT 1.E6
4950 ISOURCE One:        DAT 1.
4960 ISOURCE Zero:       DAT 0.
4970 ISOURCE !
4980 ISOURCE SUB
5000 ISOURCE Find_vel:   LDA =Ns                  ! Get number of samples.
5010 ISOURCE LDB =Ns_par
5020 ISOURCE JSM Get_value
5030 ISOURCE LDA =Array
5040 ISOURCE LDB =Data_par
5050 ISOURCE JSM Get_info
5060 ISOURCE LDA =Sdev
5070 ISOURCE LDB =Sdev_par
5080 ISOURCE JSM Get_value
5090 ISOURCE LDA =Df1
5100 ISOURCE LDB =Df1_par
5110 ISOURCE JSM Get_value
5120 ISOURCE LDA =Df2
5130 ISOURCE LDB =Df2_par
5140 ISOURCE JSM Get_value
5150 ISOURCE LDA =Df3
5160 ISOURCE LDB =Df3_par
5170 ISOURCE JSM Get_value
5180 ISOURCE LDA =Theta
5190 ISOURCE LDB =Theta_par
5200 ISOURCE JSM Get_value
5210 ISOURCE LDA =Nub
5220 ISOURCE LDB =Nub_par
5230 ISOURCE JSM Get_value
5240 ISOURCE LDA =Numix1
5250 ISOURCE LDB =Numix1_par
5260 ISOURCE JSM Get_value
5270 ISOURCE LDA =Numix2
5280 ISOURCE LDB =Numix2_par
5290 ISOURCE JSM Get_value
5300 ISOURCE LDA =Numix3
5310 ISOURCE LDB =Numix3_par
5320 ISOURCE JSM Get_value
5330 ISOURCE ! The loop headed by Get_freq is repeated three times to get
5340 ISOURCE ! the count-to-frequency conversion factors (which depend on
5350 ISOURCE ! the range) for U, V, and W. Whenever a loop is controlled by

51
the variable "Count", the loop contains operations which are
the same for U,V, and W.

LDA =0
STA Count

Get_freq: LDA Count

LDB Ns

MPY

STA Elementd

LDA =Int

LDB =Arrayd

JSM Get_element

LDA Int

LDB =15360

AND B

SAR 10

TCA

LDB =15

LDB =Range1

LDB =Count

STA B,1

LDB =1

SZA Loopend

Loop: SBL 1

DSZ A

JMP Loop

Loopend: STB Int

LDA =Int

STA Oper_1

LDB =Yvar

LDA =Result

JSM Int_to_rel

LDA =3.2E4

LDB =Yvar

XFR 4

STB Oper_1

LDA =Yvar

STA Oper_2

LDA Count

STA Result

JSM Rel_math

LDA =3

ISZ Count

LDA =2

LDB =147155B

JSM Rel_math

LDB =768

JMP Continue

JMP Get_freq

LDA =Array1

LDB =768

CLR 16

ADA =16

DSZ B

JMP Continue

Zero out the entire count-to-
velocity conversion table so that
It must be recalculated for each
point. (This must be done if the
mixing frequency or ranges are
changed between counts.)
Set initial values of Su, Sv, Sw, Suu, etc. to zero.

Clear the areas of memory which will be used to hold the number of samples per count for the histograms.

Convert Theta from degrees to radians using the Rel_math utility.

Find the sine and cosine of Theta, and store them in the locations Sin and Cos, respectively.

The loop defined by Get_int calculates intermediate values used in converting counts to velocities. These values are the same for all samples in a point, so they can be calculated separately. The loop calculates (Nub-NumixN) and (Mill*DfN), where N is 1, 2, and 3.

Find NumixN=Nub-NumixN.

Find DfN=Mill*DfN.
This section filters the data to remove counts which are excessively far from the mean. First the section creates a list of the number of times each particular count appears. The list consists of 3 1024 word arrays (called Bin_u, Bin_v, and Bin_w) where the address of each word corresponds to the count, and the value of the word indicates the number of times that particular count has appeared.

This section filters the data to remove counts which are excessively far from the mean. First the section creates a list of the number of times each particular count appears. The list consists of 3 1024 word arrays (called Bin_u, Bin_v, and Bin_w) where the address of each word corresponds to the count, and the value of the word indicates the number of times that particular count has appeared.

When $Sdev > 0$, use this section to filter the data.
The mean and standard deviation of the data are calculated. The program then throws out all the data which are farther than Sdev standard deviations from the mean. If Sdev > 0, use this section to filter the data. If not, continue with the program. Get the average of each set of counts using the information in the Bin_u arrays. First, set the average = 0.

```
LDB =Sdev
LDB =1
LDA B,I ! If Sdev > 0, use this section to filter the data. If not, continue
RZA ++2
JMP Set_begin ! with the program.
LDA =0 ! Get the average of each set of
STA Count ! counts using the information in the Bin_u arrays.
LDA =Zero ! the Bin_u arrays.
LDB Count ! First, set the average=0.
SBL 2
ADB =Avg_u
XFR 4
LDA Count
ADB =1
LDB =1024
MPY
ADB =Bin_u
STA Address
LDA =1023
LDB =1
STA I
```

Get the average of each set of counts using the information in the Bin_u arrays. First, set the average=0.

```
LDA =1 ! Get a count and convert it to a real number.
STA Oper_1
LDA =Xvar
STA Result
JSM Int_to_rel
```

LDA Address ! Use the Bin_u arrays to find out how many samples there are with that count and convert that number to a real number.
STA Oper_1 ! the number of times it appears in the data.
LDA =Yvar
STA Result
JSM Int_to_rel
```
LDA =Xvar ! Multiply the count by the number of times it appears in the data.
STA Oper_1
LDA =Yvar
STA Oper_2
STA Result
LDA =2
LDB =147037B
JSM Rel_math
```

LDB Count ! Keep a running sum of the product of the count times the number of times that it appears. Store this sum in the place we will eventually use to store the average.
LDA =Yvar
STA Oper_2
LDA =2
LDB =146721B
JSM Rel_math
DSZ Address
DSZ I
JMP Sum_count
```
Get samples:

LDA =Ns
STA Oper_1
LDA =Xvar
STA Result

Get a count and convert it to a real number.

LDA =Xvar
STA Oper
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =147155B
JSM Int_to_reI

Convert the number of samples to a real number.

LDA =Xvar
STA Oper
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =147155B
JSM Int_to_reI

Divide the sum stored in Avg_u by the number of samples to get the average value of the counts.

LDB Count
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Divide the sum stored in Avg_u by the number of samples to get the average value of the counts.

LDB Count
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math

Get the standard deviation of each set of counts using the average and the information in the Bin_u arrays. First, set the standard deviation =0.

LDA =0
STA Count
LDA =Zero
STA Count

Get a count and convert it to a real number.

LDA =Xvar
STA Oper_1
LDA Int -
STA Oper_2
LDA =Yvar
STA Result
LDA =2
LDB =146717B
JSM Rel_math
Square the difference between the average and the count we got to get the deviation from the mean.

Use the bin_u arrays to find out how many counts there are with the value we have chosen, and convert that number to a real number.

Multiply the number of counts by the deviation from the mean.

Add the product to Dev_u, which currently contains a running sum of the number of counts times the deviation from the mean of each count.

Convert the number of samples to a real number.

Divide the sum stored in Dev_u by the number of samples to get the variance of the counts.

Take the square root of the variance to get the standard deviation.
8970 ISOURCE LDB =31457B
8980 ISOURCE JSM Rel_math
8990 ISOURCE !
9000 ISOURCE LDA Count
9010 ISOURCE ADA =1
9020 ISOURCE STA Count
9030 ISOURCE ADA =-3
9040 ISOURCE SZA ++2
9050 ISOURCE JMP Get_sdev
9060 ISOURCE !
9070 ISOURCE LDA =0 ! Now use the standard deviation to
9080 ISOURCE STA Count ! filter out all the counts whose
9090 ISOURCE Dev_filtri: LDA Count ! value is more than Sdev standard
9100 ISOURCE ADA =1 ! deviations away from the mean.
9110 ISOURCE LDB =1024
9120 ISOURCE MPY
9130 ISOURCE ADA =Bin_u
9140 ISOURCE STA Address
9150 ISOURCE LDA =1023
9160 ISOURCE STA I
9170 ISOURCE !
9180 ISOURCE LDA Count
9190 ISOURCE SAL 2
9200 ISOURCE ADA =Dev_u
9210 ISOURCE STA Oper_1
9220 ISOURCE LDA =Sdev
9230 ISOURCE STA Oper_2
9240 ISOURCE LDA =Xvar
9250 ISOURCE STA Result
9260 ISOURCE LDA =2
9270 ISOURCE LDB =147037B
9280 ISOURCE JSM Rel_math
9290 ISOURCE !
9300 ISOURCE LDA =Xvar
9310 ISOURCE STA Oper_1
9320 ISOURCE LDA =Int
9330 ISOURCE STA Result
9340 ISOURCE JSM Rel_to_int
9350 ISOURCE !
9360 ISOURCE LDA Count
9370 ISOURCE SAL 2
9380 ISOURCE ADA =Avg_u
9390 ISOURCE STA Oper_1
9400 ISOURCE LDA =Offset
9410 ISOURCE STA Result
9420 ISOURCE JSM Rel_to_int
9430 ISOURCE !
9440 ISOURCE LDB Offset
9450 ISOURCE TCB
9460 ISOURCE Filtri_dev: LDA I
9470 ISOURCE ADA B
9480 ISOURCE SAM ++2
9490 ISOURCE TCA
9500 ISOURCE ADA Int
9510 ISOURCE SAP ++3
9520 ISOURCE LDA Address
9530 ISOURCE CLR I
9540 ISOURCE DSZ Address
9550 ISOURCE DSZ I
9560 ISOURCE JMP Filtri_dev
Go through this whole process of checking the deviation and filtering the data three times, once for each channel. The loop Begin is performed three times, once for U, V, and W. Each time through the data array is read, a count (a raw datum) is taken from it and converted into a velocity. The velocity is stored in U, V, or W depending on whether this is the first, second, or third iteration of the loop. The first time a particular count is encountered, the velocity corresponding to it is calculated using the intermediate values found in Get_int, and the velocity is stored in a table. If that count is found again in the data array, the corresponding velocity is looked up rather than being calculated again.

Set_begin: LDA = 0

Begin: LDA = Int

Straight: STR Int

! Go through this whole process of checking the deviation and filtering the data three times, once for each channel.

The loop Begin is performed three times, once for U, V, and W. Each time through the data array is read, a count (a raw datum) is taken from it and converted into a velocity. The velocity is stored in U, V, or W depending on whether this is the first, second, or third iteration of the loop. The first time a particular count is encountered, the velocity corresponding to it is calculated using the intermediate values found in Get_int, and the velocity is stored in a table. If that count is found again in the data array, the corresponding velocity is looked up rather than being calculated again.

Set_begin: LDA = 0

Begin: LDA = Int

Straight: STR Int

Figure out which element of the data array we want to pick up.

Get a raw datum from the data array.

Strip off the first six bits of the raw data word.

See if Count = Check. If true, use the modified data word as an index. If not, use 11324 minus the data word.

Store the count we have gotten.

Now look in the appropriate part of the arrays created by the filter section.

If the word corresponding to the count is zero, the count was filtered out and should be ignored. Ignore these three counts, go back and do the next three.
10170 ISOURCE Goodcount: LDA Count | Now, use Count to find out
10180 ISOURCE LDB =4096 | which lookup table array
10190 ISOURCE MPY | we want to use, and use the
10200 ISOURCE LDB Int | count we got from the data array
10210 ISOURCE ADB =-1 | to find exactly where in the
10220 ISOURCE SBL 2 | table we want to go.
10230 ISOURCE ADA B
10240 ISOURCE ADA =1
10250 ISOURCE ADB =Array1
10260 ISOURCE STA Address
10270 ISOURCE LDB =Int2
10280 ISOURCE XFR 1
10290 ISOURCE LDA Int2 | If that table entry is zero,
10300 ISOURCE SZA Calculate | calculate a velocity for it.
10310 ISOURCE JMP Over
10320 ISOURCE Calculate: LDA =Int
10330 ISOURCE STA Oper_1 | Convert the count into
10340 ISOURCE LDA =Yvar | a real number.
10350 ISOURCE STA Result
10360 ISOURCE JSM Int_to_rel
10370 ISOURCE STB Oper_2
10380 ISOURCE LDA =R1 | Divide the range we found
10390 ISOURCE ADB Offset | earlier by the count to get a
10400 ISOURCE STA Oper_1 | frequency.
10410 ISOURCE LDA =Xvar
10420 ISOURCE STA Result
10430 ISOURCE LDA =2
10440 ISOURCE LDB =147155B
10450 ISOURCE JSM Rel_math
10460 ISOURCE !
10470 ISOURCE LDA =Numix1 | Find (Num-NumixN)-FrequencyN.
10480 ISOURCE ADB Offset
10490 ISOURCE STA Oper_1
10500 ISOURCE LDA =Xvar
10510 ISOURCE STA Oper_2
10520 ISOURCE LDA =Yvar
10530 ISOURCE STA Result
10540 ISOURCE LDA =2
10550 ISOURCE LDB =146717B
10560 ISOURCE JSM Rel_math
10570 ISOURCE !
10580 ISOURCE LDB Count | If we are calculating U, reverse
10590 ISOURCE ADB =-1 | the sign of (Num-NumixN)-FrequencyN.
10600 ISOURCE SBM ++2 | so as to reverse the sign of U.
10610 ISOURCE JMP Samesign | Leave V and Wv alone.
10620 ISOURCE LDA =Zero
10630 ISOURCE STA Oper_1
10640 ISOURCE LDA Result
10650 ISOURCE STA Oper_2
10660 ISOURCE LDA =Xvar
10670 ISOURCE STA Result
10680 ISOURCE LDA =2
10690 ISOURCE LDB =146717B
10700 ISOURCE JSM Rel_math
10710 ISOURCE Samesign: NOP
10720 ISOURCE !
10730 ISOURCE LDA Result | Find Velocity=(Num-NumixN)
10740 ISOURCE STA Oper_1 | -(FrequencyN)*(Mill*DfN)
10750 ISOURCE LDA =Df1 | and store in a place in the
10760 ISOURCE ADB Offset | lookup table corresponding to
Over:  

LDA Address  
LDA =-1  
STA Result  
LDA =2  
LDB =147037B  
JSM Rel_math  

LDA Address  
LDA =-1  
the lookup table to U, V, or W,  
LDA =U  
as appropriate.  
LDB =147a37B  
JSM Rel_math  

LDA Address  
LDA =-3  
Transfer the velocity from  
Have U, V, and W all  
LDA Count  
been calculated?  
SJZ ++2  
If not, go back again.  
JMP Begin  
STA Count  
If so, set Count = 0.  

LDA =V  
STA Oper_1  
LDA =Cos  
STA Oper_2  
LDA =Xvar  
STA Result  
LDA =2  
LDB =147037B  
JSM Rel_math  
LDA Result  
STA Oper_2  
LDA =W  
STA Oper_1  
LDA =Yvar  
STA Result  
LDA =2  
LDB =146717B  
JSM Rel_math  
LDA Result  
STA Oper_1  
LDA =Sin  
STA Oper_2  
LDA =W  
STA Result  
LDA =2  
LDB =147155B  
JSM Rel_math  

LDA =V  
STA Oper_1  
LDA =Cos  
STA Oper_2  
LDA =Xvar  
STA Result  
LDA =2  
LDB =146717B  
JSM Rel_math  
LDA Result  
STA Oper_1  
LDA =Sin  
STA Oper_2  
LDA =W  
STA Result  
LDA =2  
LDB =147155B  
JSM Rel_math  

Now take running sums of U, V, W, and several products  
of these velocities. The sums are taken using the utility  
"Add". The sums are calculated in an unusual sequence in  
order to reduce the number of program steps needed to cal-
culate them.
LDA =Su  ! Find Su=Su+U
LDB =U
JSM Add
LDA Oper_2  ! Find U*U
STA Oper_1
LDA =Uu
STA Result
LDA =2
LDB =147037B
JSM Rel_math
LDA =Suu  ! Find Suu=Suu+(U*U)
LDB Result
JSM Add
LDA =U  ! Find U*U*V
STA Oper_1
LDA =Xvar
STA Result
LDA =2
LDB =147037B
JSM Rel_math
LDA =Suuv  ! Find Suuv=Suuv+(U*U*V)
LDB Result
LDA =2
LDB =147a37B
JSM Rel_math
LDA =Suuw  ! Find Suuw=Suuw+(U*U*W)
LDB Result
JSM Add
LDA =Su  ! Find Su=Su+V
LDB =V
JSM Add
LDA =U  ! Find U*V
STA Oper_1
LDA =Uv
STA Result
LDA =2
LDB =147037B
JSM Rel_math
LDA =Suv  ! Find Suv=Suv+(U*V)
LDB Result
JSM Add
LDA =U  ! Find U*V
STA Oper_1
LDA =Uv
STA Result
LDA =2
LDB =147037B
JSM Rel_math
LDA =Suv  ! Find Suv=Suv+(U*V)
LDB Result
JSM Add
LDA =V  ! Find U*V*V
STA Oper_1
11970 ISOURCE LDA =Xvar
11980 ISOURCE STA Result
11990 ISOURCE LDA =2
12000 ISOURCE LDB =1470373
12010 ISOURCE JSM Rel_math
12020 ISOURCE !
12030 ISOURCE LDA =Suuv
12040 ISOURCE LDB Result
12050 ISOURCE JSM Add
12060 ISOURCE !
12070 ISOURCE LDA =Uv
12080 ISOURCE STA Oper_1
12090 ISOURCE LDA =W
12100 ISOURCE STA Oper_2
12110 ISOURCE LDA =Xvar
12120 ISOURCE STA Result
12130 ISOURCE LDA =2
12140 ISOURCE LDB =1470373
12150 ISOURCE JSM Rel_math
12160 ISOURCE !
12170 ISOURCE LDA =Suuv
12180 ISOURCE LDB Result
12190 ISOURCE JSM Add
12200 ISOURCE !
12210 ISOURCE LDA =V
12220 ISOURCE STA Oper_1
12230 ISOURCE LDA =Xvar
12240 ISOURCE STA Result
12250 ISOURCE LDA =2
12260 ISOURCE LDB =1470373
12270 ISOURCE JSM Rel_math
12280 ISOURCE !
12290 ISOURCE LDA =Svv
12300 ISOURCE LDB Result
12310 ISOURCE JSM Add
12320 ISOURCE !
12330 ISOURCE LDA =Sw
12340 ISOURCE LDB =W
12350 ISOURCE JSM Add
12360 ISOURCE !
12370 ISOURCE LDA =U
12390 ISOURCE STA Oper_1
12400 ISOURCE LDA =Xvar
12410 ISOURCE STA Result
12420 ISOURCE LDA =2
12430 ISOURCE LDB =1470373
12440 ISOURCE JSM Rel_math
12450 ISOURCE !
12460 ISOURCE LDA =Suw
12470 ISOURCE LDB Result
12480 ISOURCE JSM Add
12490 ISOURCE !
12500 ISOURCE LDA =W
12510 ISOURCE STA Oper_1
12520 ISOURCE LDA =V
12530 ISOURCE STA Oper_2
12540 ISOURCE LDA =Xvar
12550 ISOURCE STA Result
12560 ISOURCE LDA =2
12570 ISOURCE LDB =147037B
12580 ISOURCE JSM Rel_math
12590 ISOURCE !
12600 ISOURCE ! LDA =Suu  ! Find Suw=Suu+(V*W)
12610 ISOURCE LDB Result
12620 ISOURCE JSM Add
12630 ISOURCE !
12640 ISOURCE ! LDA =U  ! Find W*W
12650 ISOURCE STA Oper_1
12660 ISOURCE STA Oper_2
12670 ISOURCE LDA =Xvar
12680 ISOURCE STA Result
12690 ISOURCE LDA =2
12700 ISOURCE LDB =147037B
12710 ISOURCE JSM Rel_math
12720 ISOURCE !
12730 ISOURCE ! LDA =Suu  ! Find Suw=Suu+(W*W)
12740 ISOURCE LDB Result
12750 ISOURCE JSM Add
12760 ISOURCE !
12770 ISOURCE ! LDA =U  ! Find U*W*W
12780 ISOURCE STA Oper_1
12790 ISOURCE LDA =Uu
12800 ISOURCE STA Result
12810 ISOURCE LDA =2
12820 ISOURCE LDB =147037B
12830 ISOURCE JSM Rel_math
12840 ISOURCE !
12850 ISOURCE ! LDA =Suww  ! Find Suww=Suww+(U*W*W)
12860 ISOURCE LDB Result
12870 ISOURCE JSM Add
12880 ISOURCE !
12890 ISOURCE DSZ I  ! Continue to calculate running
12900 ISOURCE JMP Begin  ! sums until out of samples.
12910 ISOURCE !
12920 ISOURCE ! Now place the finished summs in the COMMON region so that
12930 ISOURCE ! the BASIC program has access to them, and then return to
12940 ISOURCE ! the BASIC program.
12950 ISOURCE Replace: LDA =Range1
12960 ISOURCE LDB =Ran1_par
12970 ISOURCE JSM Put_value
12980 ISOURCE LDA =Range2
12990 ISOURCE LDB =Ran2_par
13000 ISOURCE JSM Put_value
13010 ISOURCE LDA =Range3
13020 ISOURCE LDB =Ran3_par
13030 ISOURCE JSM Put_value
13040 ISOURCE LDA =Su
13050 ISOURCE LDB =Su_par
13060 ISOURCE JSM Put_value
13070 ISOURCE LDA =Su
13080 ISOURCE LDB =Su_par
13090 ISOURCE JSM Put_value
13100 ISOURCE LDA =Su
13110 ISOURCE LDB =Su_par
13120 ISOURCE JSM Put_value
13130 ISOURCE LDA =Suu
13140 ISOURCE LDB =Suu_par
13150 ISOURCE JSM Put_value
13160 ISOURCE LDA =Suv

64
LDB =Svv_par
J5M Put value
LDA =Suv
LDB =Suv_par
J5M Put value
LDA =Suv
LDB =Suv_par
J5M Put value
LDA =Suv
LDB =Suv_par
J8M Put value
LDA =Suv
LDB =Suvv_par
J8M Put value
LDA =Suvv
LDB =Suvv_par
J8M Put value
LDA =Suvv
LDB =Suvw_par
J8M Put value
LDA =Suvw
LDB =Suvw_par
J8M Put value
LDA =Suvw
LDB =Ngs_par
J8M Put value
LDA =Ngs
LDB =Ngs_par
J8M Put value
LDA =Bin
Transfer the contents of Bin_n to the common area without using the slow HP-supplied external subroutines.
LDB =192
LDB =16
STB Count
XFR 16
ADA Count
ADB Count
DSZ I
JMP Transfer
JMP Transfer
LIT 200
The utility "Add" is used to add up the running sums.
ISZ UtIcount
STB Oper_1
STB Oper_2
STB Result
LDA =2
LDB =146721B
JSM Rel_math
DSZ UtIcount
RET 1
JSM UtIend
13770 ISOURCE !
13780 ISOURCE

END Find_vel
ACKNOWLEDGEMENTS

We are grateful to Dr. Oktay Ozcan for help in the initial design of the system and to Dr. Dennis Johnson for many helpful comments. This work was supported by the Fluid Dynamics Research Branch, NASA Ames Research Center under Grant NCC-2-294.
REFERENCES


5. OPERATOR'S MANUAL FOR NASA 3D-LDV COMPUTER INTERFACE, NASA Ames Research Center, Writing Assoc., Sunnyvale, CA, July 1, 1982.

## TABLE 1

**MACRODYNE RANGE SETTINGS**

<table>
<thead>
<tr>
<th>MANTISA</th>
<th>1</th>
<th>500</th>
<th>1024</th>
</tr>
</thead>
<tbody>
<tr>
<td>RANGE</td>
<td>RANGE</td>
<td>RANGE</td>
<td>RANGE</td>
</tr>
<tr>
<td>0</td>
<td>32 GHz</td>
<td>64 MHz</td>
<td>31.28 MHz</td>
</tr>
<tr>
<td>1</td>
<td>16 &quot;</td>
<td>32 &quot;</td>
<td>15.64 &quot;</td>
</tr>
<tr>
<td>2</td>
<td>8 &quot;</td>
<td>16 &quot;</td>
<td>7.20 &quot;</td>
</tr>
<tr>
<td>3</td>
<td>4 &quot;</td>
<td>8 &quot;</td>
<td>3.91 &quot;</td>
</tr>
<tr>
<td>4</td>
<td>2 &quot;</td>
<td>4 &quot;</td>
<td>1.95 &quot;</td>
</tr>
<tr>
<td>5</td>
<td>1 &quot;</td>
<td>2 &quot;</td>
<td>977 kHz</td>
</tr>
<tr>
<td>6</td>
<td>500 MHz</td>
<td>1 &quot;</td>
<td>488 &quot;</td>
</tr>
<tr>
<td>7</td>
<td>250 &quot;</td>
<td>500 kHz</td>
<td>244 &quot;</td>
</tr>
<tr>
<td>8</td>
<td>125 &quot;</td>
<td>250 &quot;</td>
<td>122 &quot;</td>
</tr>
<tr>
<td>9</td>
<td>62.5 &quot;</td>
<td>125 &quot;</td>
<td>61 &quot;</td>
</tr>
<tr>
<td>10</td>
<td>31.25 &quot;</td>
<td>62.5 &quot;</td>
<td>30 &quot;</td>
</tr>
<tr>
<td>11</td>
<td>15.62 &quot;</td>
<td>31.25 &quot;</td>
<td>15 &quot;</td>
</tr>
<tr>
<td>12</td>
<td>7.81 &quot;</td>
<td>15.62 &quot;</td>
<td>7.6 &quot;</td>
</tr>
<tr>
<td>13</td>
<td>3.90 &quot;</td>
<td>7.81 &quot;</td>
<td>3.8 &quot;</td>
</tr>
<tr>
<td>14</td>
<td>1.95 &quot;</td>
<td>3.90 &quot;</td>
<td>1.9 &quot;</td>
</tr>
</tbody>
</table>
Fig. 1  Schematic of the optics table layout.
Fig. 2
Schematic of the transmitting optics.

MICRO
POTRIZATION
RTORATOR

LEN S f = 15 in.

LEN S f = 15 in.

DICHROIC FILTER
(SPLITS GREEN)
(LINE)

FOUR BEAM
MATRIX FROM
OPTICS TARE FL
Fig. 3 Details of the probe volumes in the 3-D system.
Fig. 4  Schematic of the receiving optics.
Fig. 5 Evaluation of the effective probe length.

\[ L_{\text{eff}} = \frac{df_1}{f_2 \sin \theta} \]

Fig. 6 Schematic of the signal processing electronics.
Fig. 7  Computer interface simplified block diagram.
Fig. 8 Block diagram for data acquisition program.
Fig. 9

Block diagram for data reduction program.
Fig. 10 NASA LDV-A/D computer interface connections and settings.
(a) Overall schematic

(b) Details of boundary layer trips and coordinate system

Fig. 11 Experimental rig.
Fig. 12 Secondary velocity plots.
Fig. 13  Comparison of X-wire and LDV measurements in a vortex/mixing layer interaction.
End of Document