An Experimental Investigation of the Fluctuating Wall Shear Stresses in an Eight Degree Conical Diffuser

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in
Department of Mechanical Engineering

by

Robert Wayne Derksen

Winnipeg, Manitoba

AN EXPERIMENTAL INVESTIGATION OF THE FLUCTUATING WALL SHEAR STRESSES IN AN EIGHT DEGREE CONICAL DIFFUSER

BY

ROBERT WAYNE DERKSEN

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

DOCTOR OF PHILOSOPHY

I hereby declare that I am the sole authour of this thesis.

I authourize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Robert Wayne Derksen

I further authourize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Robert Wayne Derksen

Abstract

This thesis reports on a set of experimental measurements of the fluctuating wall shear stress in an 8° conical diffuser. The diffuser was fed by fully developed pipe flow at a Reynolds number of 130,000. Hot-film flush mounted probes were used to measure the fluctuating wall shear stresses.

Various single point statistical measurements were made at various axial locations through the diffuser. These measurements were the central moments up to 6th order, the spectra, autocorrelations, probability density functions and a conditional sampling measurement proposed by Zaric. Although some of the measurements showed some variation with axial position in the diffuser, the bulk of the measurements indicated that the turbulence structure in the viscous sublayer was independent of the location and hence the pressure gradient. The spectra were found to have a universal form when normalized by τ' and an outer time scale formed by the local area weighted mean velocity and the diameter. The outer time scale and τ' were functions of location in the diffuser. The fluctuating shear stress intensity was found to increase in the downstream direction, however the increase appears to be a delayed response to the imposition of the pressure gradient.

Two point measurements were made of the joint moments up to 4th order and the joint probability density functions. The joint moments were asymmetric with slightly greater correlation with downstream points. The axial extent of the flow structure was of the order of 5 or 6 local diameters. Due to an overly large initial displacement, the transverse correlations were slightly negative for all measured points. This negative character is typical and indicates that the transverse scale of the turbulence was less than one local diameter.

Acknowledgements

I would like to thank all of the people who have encouraged and assisted me in my Ph. D. program: my mother, who has been very supportive and has provided aid and assistance that has made my completion of this project possible; Dr. R. S. Azad for his guidance, encouragement and extremely helpful assistance; Laura Ozimek for proof reading this thesis and correcting the spelling and grammatical errors; my Ph. D. advisory committee, for having guided my progress and having read this thesis.

I would like to acknowledge, and thank the granting bodies for the financial support of the National Sciences and Engineering Research Council Scholarship, the University of Manitoba Fellowship and the Alfred Rea Tucker Memorial Scholarship.

Nomenclature

a	$\lambda^2\int_0^1 \eta f(\eta)d\eta$
A	a constant
b	$\lambda^3\int_0^1 \eta^2 f(\eta)d\eta$
B	a constant
C	a constant
Cf	skin friction coefficient
Cp	heat capacity
d	Preston tube diameter
D	a constant, or the diameter of the inlet pipe
D_0	local diameter in the diffuser
E_1	principal mean axial shear strain rate
E_2	principal mean radial strain rate
f	frequency
fc	cut-off frequency of spectral measurements
$f(\eta)$	the non-dimensional temperature distribution function in a laminar boundary layer as a function of η
F( )	flatness factor
h	thickness of the viscous sublayer
JPDF	joint probability density function
i	$\sqrt{-1}$
I	electric current, or integral time scale
Ic	corrected integral time scale
k	thermal conductivity

K_5	super skewness (the nondimensional 5th moment)
K_6	super flatness (the nondimensional 6th moment)
L	length of the hot-film
L_1	heated length of the substrate upstream from the hot-film
L_{eff}	effective length of the hot-film
P	static pressure
P_p	Preston pressure
Pr	Prandtl number
PDF( )	probability density function
Q_w	wall heat flux
R	resistance
Re	pipe Reynolds number, DU_b/ν
RMS	root mean square
R( )	autocorrelation function
S( )	skewness
SD( )	standard deviation
T	temperature
T_0	local time scale, D_0/Um
T_1	ambient fluid temperature
T_w	wall temperature
u	fluctuating axial velocity component
u_{mp}	the most probable value of u
u_*	friction velocity
u_+	non-dimensional mean velocity, $\bar{U}/u_*$
$\bar{U}$	mean axial velocity component
U_b	the bulk velocity in the pipe
Um	area weighted local bulk velocity

v	fluctuating radial velocity component
x	axial position
x^*	non-dimensional argument of P_p for Preston tube
y	distance from a wall
y_+	non-dimensional distance from a wall, yu_*/ν
y^*	non-dimensional function of τ for Preston tube
Z_0	non-dimensional shear stress τ_0/τ_0'
Z_1	non-dimensional shear stress, τ_1/τ_1'
β	$(\omega\delta^2/\kappa)^{1/2}$
γ	intermittancy or probability
δ_T	thermal boundary layer thickness
Δ	dimensionless pressure gradient parameter, $(\nu/\rho u_*^3)dP/dx$
ΔP	contraction cone pressure drop
Δt	time shift for autocorrelation
ΔT_0	Temperature difference between the wall and ambient fluid, T_W-T_1
$\Delta\theta$	circumferential shift in position
η	non-dimensional distance from a wall y/δ_T
θ	angular or circumferential position
κ	thermal diffusivity
λ	temperature profile shape parameter, $Q_W\delta_T/k(T_W-T_0)$
μ	dynamic viscosity
ν	kinematic viscosity
Ξ	non-dimensional axial distance, x_1-x_0/D_0
π	3.1459...
ρ	mass density
τ	fluctuating wall shear stress

τ'	RMS of the fluctuating wall shear stress
$\bar{\tau}$	mean wall shear stress
$\bar{\tau}_m$	measured wall shear stress
$<\tau>$	wall shear stress during a conditional event
ω	circular frequency, $2\pi f$

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter I

Introduction

1.1 PREAMBLE

The intention of the work described in this thesis was to examine the characteristics of the turbulent flow in the immediate vicinity of the wall in an 8° conical diffuser. This was done by using experimential measurements of the fluctuating wall shear stress. Similar studies have been made in zero pressure gradient boundary layer flow and in pipe flow. These measurements, it is hoped, will reveal the characteristics of the viscous sublayer without the difficulties of velocity measurement. Hot-wire measurements near the wall, within the viscous sublayer, are complicated by a complex thermal interaction with the wall and an unknown aerodynamic effect in the flow. Laser doppler anemometry is also more difficult near a solid surface because of seeding difficulties and resolution problems. For this reason, hot-film shear stress gauges were selected to record the instantaneous wall shear stresses. Hopefully the new shortcomings that are incured with this method are less severe than those for the other methods.

The 8° conical diffuser was selected because it was relatively easy to make, it was one of the simpler representations of an adverse pressure gradient flow and because of our familiarity with this flow at the University of Manitoba. This diffuser has the largest efficiency of all of the conical diffusers, and because of this the flow has the largest pressure gradient (at a given axial position) without separating. This is also an important flow in terms of industrial applications.

Before starting these experiments, very little was known about the behaviour of the viscous sublayer in an adverse pressure gradient. Hence the objective was to experimentally explore the behaviour there.

1.2 FLOW IN AN 8° CONICAL DIFFUSER

The turbulent flow in an 8° conical diffuser has been extensively studied. The flow under consideration has fully-developed pipe flow as an inlet condition and a free exhaust at the outlet. The mean axial velocity distribution, from Derksen and Azad [1980], is shown in figure 1.1. The velocity profile shows a strong similarity to pipe flow for roughly the first 1/3 of the diffuser's length, which develops into an inflectional profile with a high degree of curvature in the core of the final 1/3 of the diffuser's length. The velocities were shown to be Reynolds number independent when they were non-dimensionalized by an appropriate velocity scale such as the bulk velocity in the pipe or the maximum cross-sectional velocity. A more revealing view is given in figure 1.2, again from Derksen and Azad, which shows the mean strain rate field in the diffuser. In this figure a local maxima in the mean strain rate field can be observed in the later regions of the diffuser. This maxima is located on a line that diverges from the wall at about 1/3 of the diffuser length and moves toward the centerline as one moves downstream. It is believed that other than wall jets this feature is unique among wall bounded flows. The flow in the latter part of the diffuser displays characteristics of a jet in the core, a boundary layer near the wall and a mixing layer between these regions.

Measurements of the second order moments, the normal and tangential Reynolds stresses, are given by Okwoubi and Azad [1973], and Arora and Azad [1980a,1981]. Okwoubi and Azad found that these quantities were Reynolds number independent when nondimensionalized by the mean velocities. These moments can be shown to have a maximum near the maxima in the mean strain rate.

Arora and Azad [1980a] found that the production and dissipation were the same order of magnitude in the diffuser although not equal at every cross-section. At the inlet, turbulent kinetic energy was produced in excess of the dissipation. Near the walls, dissipation was larger than production. Thus turbulent kinetic energy was transported radially and downstream. The pressure transport term of the turbulent energy balance was found to be dominant.

Measurements of the skewness and flatness factors of the axial fluctuating velocity components were described by Arora and Azad [1981] and Azad and Hummel [1979]. They found that the skewness was zero and the flatness was minimum along a line similar to the maxima in the mean strain rate. The skewness was negative in the core region of the flow and positive in the wall region. This indicates a much more complex flow structure than that indicated by assuming the flow is a mixture of a jet, a mixing layer and a boundary layer.

Arora and Azad [1980a] and Azad and Hummel [1979] found that the spectrum of u, the axial component of the fluctuating velocity, displayed the typical -5/3 law. On examining the structure functions, Azad and Hummel [1981] found the existence of an inertial subrange for a one decade separation distance. They concluded that the predictions of Kolmogorov's origional theory had validity even in this flow. Arora and Azad [1980b] found that in the core of the diffuser isotropic vorticity theory was valid.

This has been a very brief overview of the characteristics of the flow in an 8° conical diffuser. The flow can be seen to be unique and of interest to the study of turbulence. At this time the full picture of the flow characteristics is not complete. One area where data has been missing is near the wall.

1.3 REVIEW OF THE VISCOUS SUBLAYER

Generally the flow near a wall is determined by wall parameters such as the wall shear stress, the kinematic viscosity and the distance from the wall. This region is called the 'inner' layer and it is right next to the wall. The inner layer comprises roughly 0.1 to 0.2 times the boundary layer thickness of a zero pressure gradient boundary layer. Additionally the inner layer may be subdivided into three layers, a fully turbulent layer where the total stress is dominated by the Reynolds stress, a buffer layer where the Reynolds stress and the mean viscous stress are both important and a viscous sublayer where the mean viscous stress is dominant. It is the viscous sublayer that we will concentrate on.

Usually the viscous sublayer is defined as $y_+<5$ where $y_+=yu_*/\nu$. For air, this layer is very thin, of the order of 0.5 mm and less. The existence of a thin layer, with a linear mean velocity distribution, was first hypothesized by Prandtl. Its existence was confirmed by Stanton, Marshall and Bryant [1920], who described this layer as 'laminar' because of the linear velocity distribution. This resulted in the old terminology of the laminar sublayer. Unfortunately it was assumed that the flow there was truly laminar and that the fluid moved in rectilinear motion. By examining the motion of dust particles under a microscope, Fage and Townend [1932] found that the flow was distinctly not rectilinear.

Scientifically, the viscous sublayer represents an interesting interaction between the turbulence in the fluid and the wall due to the action of viscosity. Its mechanism must be understood if a universal theory of turbulence is ever going to be developed. To this effort many theoretical models have been developed.

1.3.1 Theories of the viscous sublayer

One of the earliest theories is the discontinuous film model. It has been expounded by Einstein and Li [1956] and Hanratty [1956]. In this theory it is supposed that a cyclic pattern of events occurs. At some initial time the temperature or concentration of the fluid is assumed to equal the free-stream value right up to the wall, the velocity at the wall is assumed to equal the mean velocity at some distance from the wall. After this time a layer grows due to molecular diffusion. Then at some point in time the layer breaks down due to some unspecified non-linear effect and the process starts over. The calculation of the mean mass transfer, heat transfer or shear stress is critically dependent upon the period of this process and in the case of the shear stress the initial 'wall' velocity. This point has been the focus of criticism by opponents of this theory. The violation of the no-slip and continuity conditions has also been criticized.

The Navier-Stokes equations have been the origins of many sublayer theories. Sternberg [1962,1965] used the equations for the fluctuating velocity components. He truncated the equations by arguing that, near the wall, the convection terms and the non-linear terms were negligible. This resulted in a system of linear differential equations that were coupled by the continuity equation. The velocity was then expressed as being periodic in time and the streamwise direction, which is equivalent to a Fourier transform. The variation in the normal direction was then calculated. By using spectra that were measured at points away from the wall, he then computed the variation of the turbulence intensity and spectra nearer to the wall. His spectra tended to be too low in the low frequency range and the quality of agreement of his intensities was very dependent on where the measured spectra came from. The treatment by Schubert and Corcos [1967] is similar to that by Sternberg except they include the convection terms and omit only the variation of the instantanious Reynolds stresses. Both of these theories assume, to quote Schubert and Corcos,

> 'at least in the coarse sense, one may view the turbulent pressure in the neighbourhood of the wall as the result of turbulence outside of this thin layer and as driving the velocity fluctuations within it.'.

Landahl [1967, 1972, 1975] proposed a more complex model than that of either Sternberg or Schubert and Corcos. In it, he proposed that a higher frequency, or wave number, patch of waves rode on the lower frequency wave. An Orr-Sommerfeld analysis was used to compute the eigenfunctions and to assess the stability of the low frequency wave. He concluded that the small scale waves could be focused and cause a breakdown of the large scale structure. His model implies that energy is transmitted from the small scale structure to the large scale structure. Generally it is believed that the energy flows the other way.

Taking the curl of the Navier-Stokes equations gives the Helmholz equations (vorticity equations) which do not contain the pressure terms. Lyatkher [1968] used this equation in its fluctuation form to develop a sublayer theory. He assumed a two dimensional turbulence and Fourier transformed the equations with respect to the streamwise distance and time. Using the transformed equations he obtained the first four derivatives of the transform with respect to y at the wall. Then he used these derivatives in a Taylor series expansion of the transform with the wall as the origin. The spectra of the wall shear stress was the only required boundary condition. For this he was able to estimate the spectra of u, v and uv, as well as the RMS values of these terms as a function of the distance from the wall.

Black [1966,1969] described a chain of events in the layer from $y_+=0$ to $y_+=50$ in which elongated ring vortices are convected along with the fluid. As the ring vortex is convected, the near wall portion moves less quickly than the portion further from the wall. Because of this the vortex is stretched. Unfortunately he assumed that a Blasius layer grows along the wall between events. This is very similar to the assumptions of Hanratty and Einstein and Li, in their discontinuous film model.

One of the simplest models is that proposed by van Driest [1956]. He noted that the amplitude of fluctuations in velocity decay exponentially with distance from the oscillating flat plate in a semi-infinite laminar flow. Because of this he proposed that the mixing length should incorporate a damping function equal to one minus an exponential decay function. His mean velocity distribution fits the measured velocity distribution very well for the entire inner layer. This law fits partly because there is an adjustable constant. Unfortunatly this is an empirical scheme and it has provided no real insight into the flow behaviour near the wall.

1.3.2 Flat plate, channel and fully-developed pipe results

An assembly of the measured shear stress intensity $(\tau'/\bar{\tau})$ is given in table 1.1. With the exception of Blinco and Simons [1974] all of the reports indicate that the intensity is constant with respect to Reynolds number, and that it has a value of about 0.25 to 0.32. It is quite possible the variation is due to differences in the measurement technique and linearization of the measuring device. The reported values of skewness and flatness are shown in table 1.2. They also appear to be independent of the Reynolds number. The measured skewness appears to be about 0.5 and the flatness about 3.0.

An example of the probability density function is given in the figure of Py and Duhamel [1972] shown in figure 1.3 It is positively skewed with what appears to be Reynolds number independence. This graph is characteristic of most other reported probability density functions.

The spectra of the mass-transfer fluctuations of Mitchell and Hanratty [1966] are shown in figure 1.4. They are proportional to the spectra of the shear stress fluctuations. When non-dimensionalized by outer variables D and U they have a universal form. This is also characteristic of the other published results.

Popovich and Hummel [1967] did an experiment in which a flash photolysis technique was used. From it they observed that very near the wall ($y_+ < 1.6$) the velocity distribution was always linear. As the distance from the wall increased the probability that a linear velocity profile extended to that point decreased. At $y_+ \simeq 20$ the probability was zero.

1.3.3 A conditional sampling routine due to Zaric

This method of conditionally sampling a signal was introduced by Zaric [1972, 1974, 1975] and was later elaborated on by van Thinh [1982]. It is based on the use of a discriminator signal formed from $(u-u_{mp})du/dt$, where u_{mp} is the most probable value of u. When the discriminator signal is above a certain value it is assumed that a sweep event is occuring and when it is below a certain value an ejection is occuring. The actual method that had to be employed was more complex in that an effort had to be made to keep the gating from switching on and off too rapidly.

Zaric [1975] concluded that his conditional sampling routine could detect the presence of the different phases of turbulent flow near a solid wall. He showed that near the wall the probability density distributions could be decomposed into three separate probability density distributions. Based on this he speculated that a 'three-fluid' model could be used to describe the flow near a wall.

TABLE 1.1

The $\tau'/\bar{\tau}$ in flat plate, pipe and channel flows.

Authour	Flow	Re	$\tau'/\bar{\tau}$	Measurement Tech.
Fortuna & Hanratty	pipe flow Aqueous sol	14000	0.32	Electro-chem.
Py	channel flow Aqueous sol.	5000 62000	0.30	Electro-chem.
Py & Duhamel	channel flow aqueous sol.	3500	0.35	Electro-chem.
Hanratty	pipe flow aqueous sol.	10000 62000	0.32	Electro-chem.
Sreenvasan & Antonia	channel flow air	15000 25400	0.25	Hot-film
Klages	pipe flow oil	7600 17900	0.20	Hot-film
Eckelmann	channel flow oil	5600 8200	0.24	Hot-film
Kreplin	channel flow oil	4800 7100	0.25	Hot-film
Kreplin & Eckelmann	channel flow oil	7700	0.25	Hot-film
Blinco & Simons	tilting flume water	4000 150000	1.2-0.2 decreasing with Re	Hot-film
Mitchell & Hanratty	pipe flow aqueous sol.	10000 70000	0.32	Electro-chem.

TABLE 1.2

The skewness and flatness of τ in equilibrium flows.

Authour	Re	$S(\tau)$	$F(\tau)$
Sreenivasan	$11.78X10^3$	0.58	3.05
& Antonia	10.34	0.52	3.04
	9.14	0.51	3.30
	8.21	0.55	3.02
	7.04	0.53	3.19
	6.05	0.53	3.10
Eckelmann	$8.2X10^3$	0.75	3.70
Py &	$3.2X10^3$	0.608	3.01
Duhamel	5.0	0.560	2.98
	7.2	0.537	2.96
	9.95	0.510	2.96
	15.0	0.502	2.95
Blinco &	$40.0X10^3$	0.9-0.6	4.3-3.0
Simons	$15.0X10^4$		

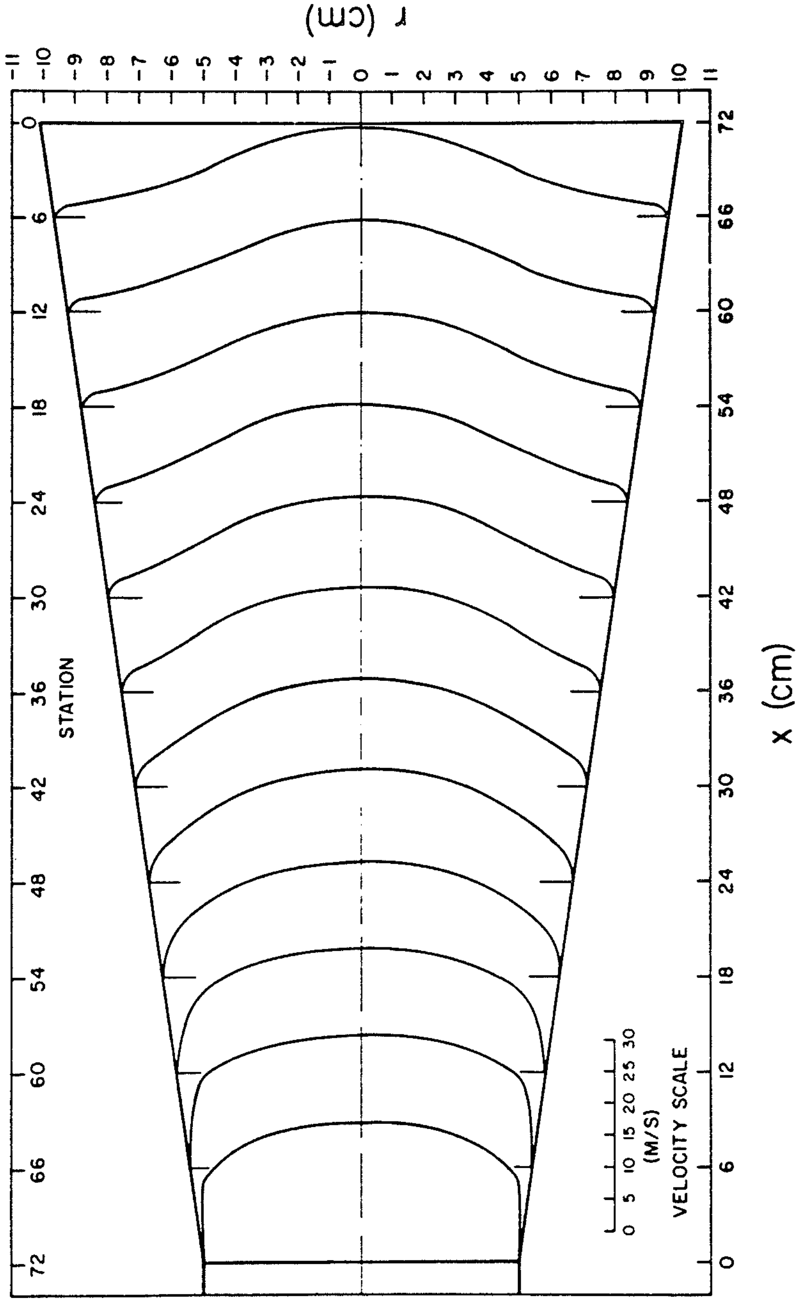


Figure 1.1: The $\overline{U}$ distribution in the 8° conical diffuser.

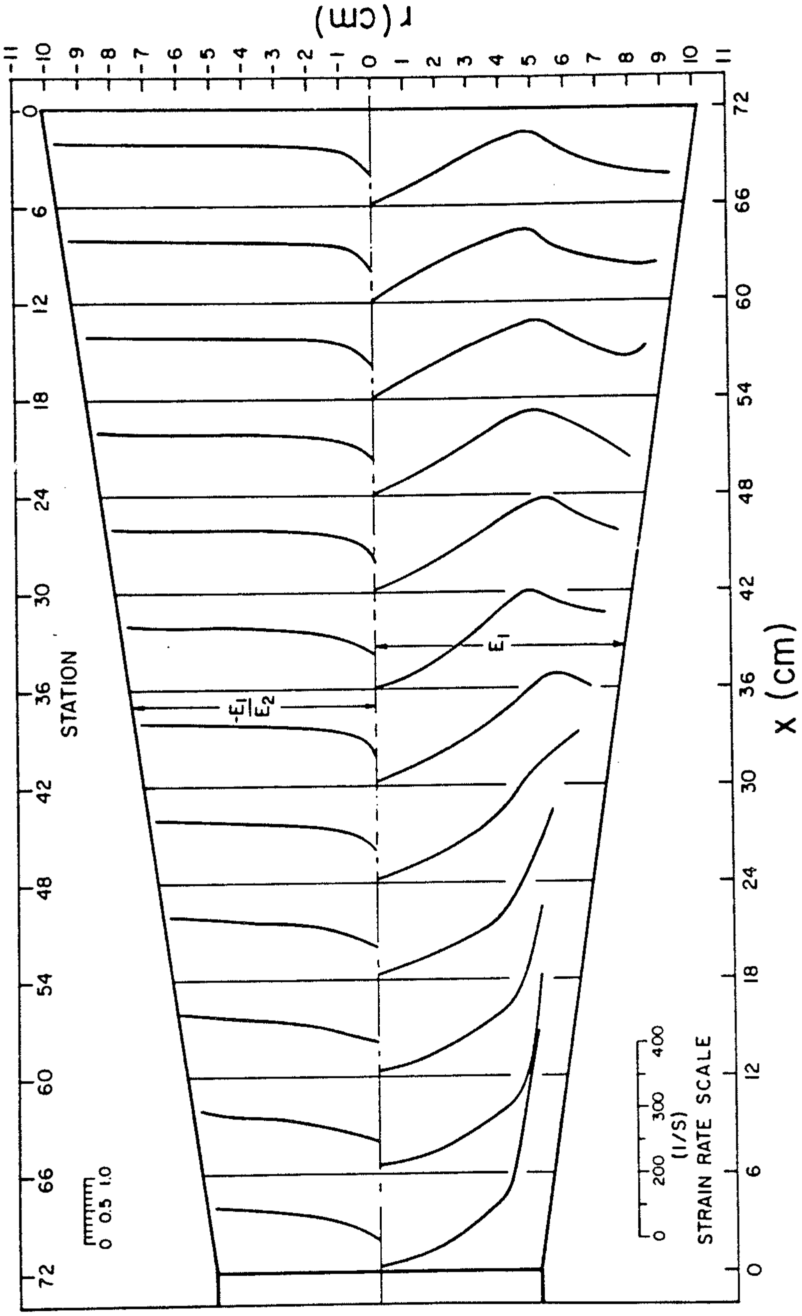


Figure 1.2: The mean strain rate field in the 8° diffuser.

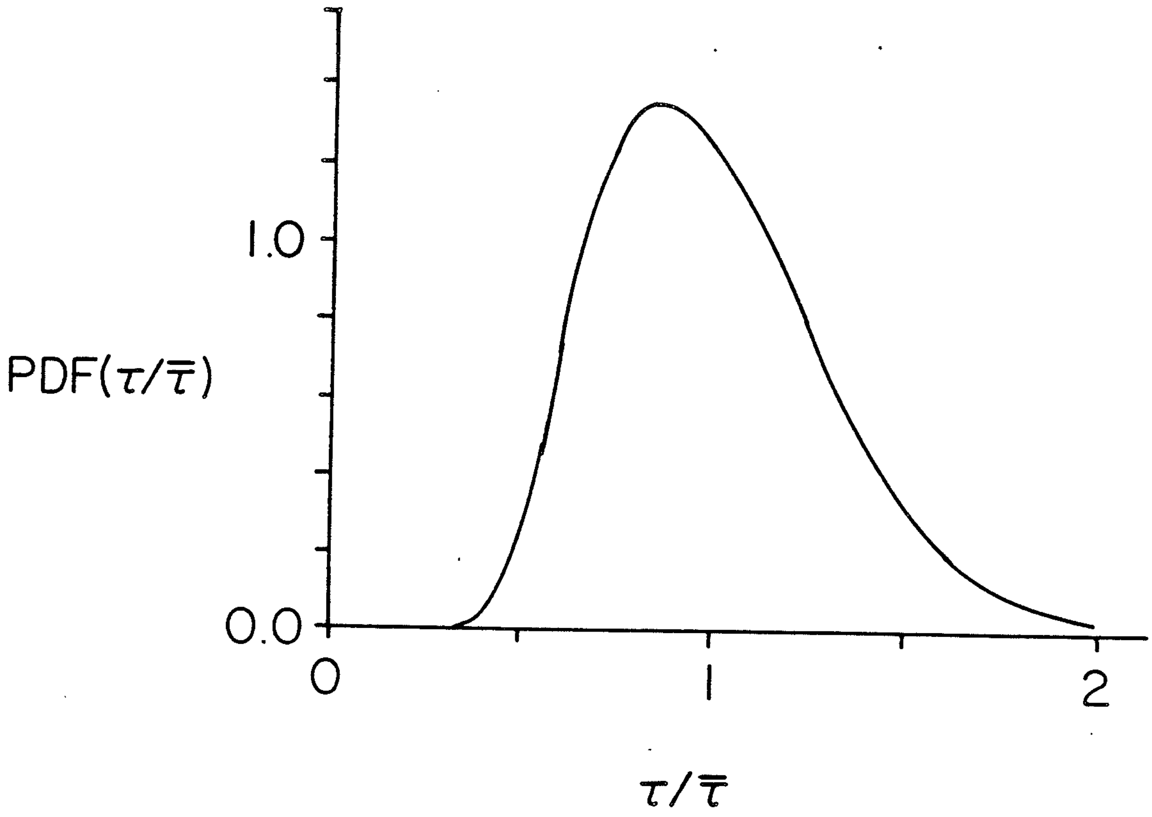


Figure 1.3: Py and Duhamel's shear stress probability densities.

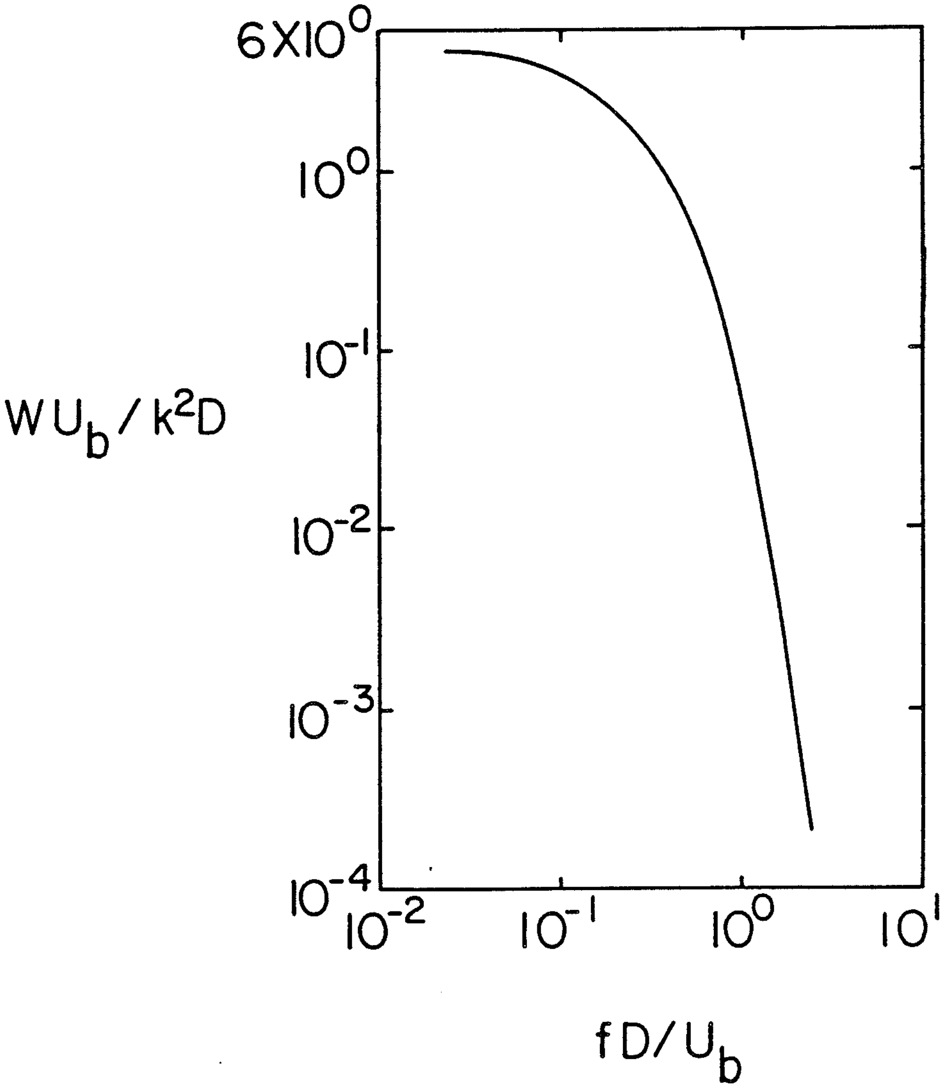


Figure 1.4: Mitchell and Hanratty's shear stress spectra.

Chapter II

Wind Tunnel and Instrumentation

2.1 THE WIND TUNNEL

The wind tunnel was a straight open return tunnel. Air at room temperature and atmospheric pressure was drawn into a radial flow fan. The fan was driven by a 25 hp DC electric motor with a continuously variable speed control. Then the air was driven through a diffuser into a settling chamber. There were 3 sets of copper wire screens in the settling chamber used to break up any large scale flow structures from the fan. Following the settling chamber, the air flowed through a contraction cone into a combination vibration isolator and flow trip. Next the air flowed through a long section of round pipe into the diffuser. The diffuser vented directly into the laboratory where the air recycled to the fan. The general layout of the wind tunnel is shown in figure 2.1

2.1.1 The laboratory and atmospheric conditions

The laboratory and atmospheric conditions do influence the performance of an open return wind tunnel. The laboratory was approximately 21 m long, 11 m wide and 4 m high with the wind tunnel located along the center of room, parallel to the long wall. There were two doors, a single door near the corner by the diffuser and a double door near the fan. Another open return wind tunnel was located along side of this tunnel. The laboratory also housed a technician's office, a PDP-11 computer, a calibration stand and a small work area. At times the laboratory could be quite active.

Environmental control in the laboratory consisted of a row of steam radiators along the floor, and two rows of cold air ducts on the ceiling. This resulted in a moderately uniform room temperature thoughout the laboratory. From day to day the temperature of the laboratory was noted to range from 22 °C to 26 °C, with most of the temperatures falling within 23 °C to 25 °C. This results in the air having a density variation of around 0.5% due to the temperature fluctuation. The barometric pressure at the instalation varied from 732 mm Hg to 746 mm Hg. Again this results in a density variation of around 0.5% due to pressure fluctuation. Thus the air density was constant to within about 1.0%. Using the tables given by Goldstein [1938] one can see that the kinematic viscosity also varies slightly, to within 2%.

The variations of the room environment can be seen to have a very minor effect on the flow. This is not true for hot-wire and hot-film applications, where special care must be taken.

2.1.2 The fan and motor

The fan was a large radial flow fan driven by a 25 hp DC electric motor. A continuously variable DC power supply was used to manually control the motor. An iron gear had been attached on the free end of the rotor. A magnet was fixed on one side of this gear and a magnetic pickup (a small tape recorder head) was fixed to the other side. When the motor turned the magnetic field lines were interruped, and by counting the pulses the motor speed could be measured. This pulse rate was used to monitor the Reynolds number of the tunnel.

This system was slightly dependent upon room temperature and the external power source. Because of this, the tunnel control settings had to be very carefully monitored or the tunnel Reynolds number would drift or fluctuate.

The fan inlet was a 1.0 m circular opening, coaxial with the fan axis, with adjustable guide vanes. The fan had a 0.8 m inside diameter and a 1.6 m outside diameter. It was composed of cambered airfoil elements that were backward inclined. The fan housing had a 33.0 cm by 70.0 cm rectangular discharge.

The diffuser between the fan and the settling chamber was of heavy sheet metal construction. It had a rectangular inlet, matching the fan outlet, and opened up to a 95.0 cm diameter circular outlet. It had an overall length of 210.0

cm. Guide vanes were installed in the diffuser to straighten and condition the flow from the fan.

2.1.3 Settling chamber and contraction cone

The settling chamber was a heavy sheet metal circular tube which mated the aforementioned diffuser. It was 2.2 m long. There were 3 sets of copper screens, with the first located 40 cm from the inlet and each one 15 cm apart. The screens were used to breakup any residual flow structures from the fan.

Following the settling chamber was an axisymmetric contraction cone. It was constructed from a stack of plywood rings that were smoothed into a gradual and curved surface. The contraction cone had a 89:1 area ratio, in a 145 cm length. A ring of static pressure taps was located just before and just after the contraction cone. These taps were connected to a Trimount inclined manometer, calibrated in inches H_2O. The calibration was from 0.00 in. H_2O to 8.00 in H_2O in divisions of 0.02 in. of H_2O. This pressure drop was used to set the tunnel speed.

2.1.4 Combined vibration isolator and flow trip

This item was made from a 10.16 cm inside diameter steel pipe, as shown in figure 2.2 . The section containing the sandpaper trip had been machined out in order to partially submerge the trip. A coarse grit sandpaper trip was used to force transition to turbulent flow immediately and thus reduce the subsequent length of pipe necessary to obtain fully developed turbulent pipe flow. The rubber section was used to reduce fan and motor vibration that was transmitted along the tunnel.

2.1.5 Pipe section

Following the combined vibration isolator and flow trip there were two straight sections of 10.16 cm ID pipe. The first pipe was 305 cm long and the next was 405 cm for a combined length of 7.1 m. This was enough to insure fully developed pipe flow at the outlet. The pipes were connected butt to butt. The inside surface had been sanded smooth with 600 grit sandpaper to insure that it was hydraulically smooth.

The pipes were supported on wheeled jack posts, and these jack posts were placed on wooden benches. The axis of the pipe was approximatly 2.1 m above the laboratory floor.

2.1.6 Fully developed flow instrument section

This section is shown in figure 2.3.It was composed of a 30 cm section of 10.16 cm pipe. A square opening had been machined into one side of this section of pipe. In this opening, an aluminium insert was installed to support a flush mounted hot-film probe. The insert was machined to fit flush to the inside surface of the pipe. In the insert a 1.2 cm hole had been drilled to hold a plastic probe holder. A 1.0 mm static pressure tap with a 0.5 mm Preston tube had been installed on the opposite side of the pipe to the opening for the hot-film.

2.1.7 The diffuser

The diffuser was an 8 ° conical diffuser. It was machined out of two blocks of clear plastic. This resulted in a joint midway along the diffuser where there was a slight indentation on the surface. The diffuser dimensions and geometry are shown in figure 2.4 .

The diffuser had 72 holes drilled into it, with two individually fitted plastic plugs for each hole. One plug was solid and the other one had a hole drilled into it to accept either a static pressure tap or a flush mount hot-film. Unfortunately the plastic plugs' fit to the diffuser surface was poor and this could have contributed unnecessary measurement error.

Table 2.1 contains a list of all 72 plugs and their coordinates. The plugs were arranged in 9 banks of 8 plugs, where each bank was located at a single axial position. The plugs in each bank were located with equal space between them.

This was the last element in the wind tunnel. The air was discharged directly into the laboratory from the diffuser.

In order to position probes in the flow a milling table had been placed at the outlet of the diffuser. It was capable of x-y motion with rotation of the table.

2.1.8 Wind tunnel operation

The Reynolds number (based on the bulk velocity in the pipe and the pipe diameter) and fan speed were measured for various cone pressure settings. The Reynolds number was proportional to the fan speed and the square root of the cone pressure. This gave a calibration of the wind tunnel in terms of either the cone pressure or the fan speed. Hence either could be used to set the operating point for the tunnel.

2.2 THE INSTRUMENTATION

Various pressure measurement devices and electronic instruments were used for these experiments. They will be described in detail in the following sections. Details on the Preston tube and hot-film shear stress methods are given in appendices. The details about the digital data acquisition and analysis are also given in appendices.

2.2.1 Pressure Based Devices

2.2.1.1 Pitot and Preston tubes

There were two Preston tubes used in these experiments. They were of inhouse manufacture and their geometry is shown in figure 2.5 . The two tube diameters were 0.55 mm and 1.075 mm. The tubes were held in holders mounted to the milling table, and the milling table was used to position the tubes.

The Preston pressure was measured with a Combist manometer. Other information that was recorded was the air stream temperature, the fan speed and the contraction cone pressure. These measurements were to serve as a calibration standard for the hot-film probes. During these measurements it was observed that the air stream temperature increased with the contraction cone pressure. The temperature of the air increased about 4 °C when the cone pressure was raised to 4.5 in H_2O. This caused the pipe section to expand about 1.0 mm and hence the Preston tube had to be repositioned for each measurement.

The Pitot tube was used to measure the velocity distribution in the diffuser at 31.55 cm from the inlet. A United Sensor USC-E-120-03 flattened impact tube was used with the pressure measured by a Betz manometer. This velocity distribution was used to obtain the volumetric flow rate for various cone pressure settings to calibrate the wind tunnel.

2.2.1.2 Manometers

Two types of manometers were used for pressure measurements, a Combist micromanometer and a Betz manometer. The Combist had a range from 0 to 30 mm H_2O, with the finest division of 0.01 mm H_2O. It is described by Bradshaw [1965]. The Betz had a range of -17 to 400 mm H_2O, with the finest division of 0.1 mm H_2O.

The response time of the Combist was the faster of the two, as the Betz has a very large reservoir. For very highly fluctuating flows this made the Combist impossible to read to the finer divisions, with a significant loss of precision. In this event the Betz is more precise provided sufficient settling time is allowed. Sometimes the Betz took 5 minutes to stablize.

2.2.2 Electronic measuring devises

2.2.2.1 Hot-film shear stress probes

The flush mount hot-film probes that were used were manufactured by DISA, model 55A92. The length of the film was 0.2 mm and its width was 1.0 mm. Note, this follows the standard flush mount shear stress probe conventions where the length is the dimension parallel to the direction of the flow and is thus the smaller dimension. The hot-films were operated at an overheat ratio of 0.4 in the constant temperature mode. This overheat ratio was selected to obtain the maximum, safe sensitivity to the shear stresses. For this probe DISA indicated that the probe has a response that is flat up to 30 kHz. This was qualitatitively confirmed by examining the square wave response. Freymuth and Fingerson [1977] have shown that the cutoff frequency of the combined hot-film and constant temperature circuit is nearly inversely proportional to the impulse response time of a square wave. This time was found to be of the order of 30 μsec.

2.2.2.2 Constant temperature units

The flush mounted hot-film probes were operated by DISA 55M01 main units with DISA 55M10 standard bridges. In operation the power supplies, DISA 55M05, were remotely located from the main units with a connection through a power cable. This was done to reduce electromagnetic noise from the power supply. With the power cable, the hum induced at the input was about 0.1 μV RMS. The noise voltage from the bridge was quoted as less than 0.19 mV RMS for a hot-film probe. The RMS voltage fluctuation due to the shear stress fluctuation was of the order of 100 mV.

2.2.2.3 Linearizers

DISA 55D10 linearizers were used in an attempt at linearizing the bridge output. With the change in the air stream temperature with the Reynolds number it was impossible to obtain a complete linearization. All that could be done was to set the linearizer for an exponent of 1/3 (the theoretical value for hot-film shear stress probes) and record the mean voltage for a few shear stress values around the measurement point to obtain the static sensitivity. For the sensitivity to the fluctuating part, the static sensitivity had to be multiplied by 4.115 as shown by Bellhouse and Schultz [1968]. This is an obviously undesireable situation, and it is highly recommended that some means of more satisfactory linearization be developed, before more work is done with the hot-film shear stess probes.

2.2.2.4 Signal conditioning and the analog tape recorder

Signal conditioning was done using DISA 55D25 auxiliary units. These units were used to force the input signal between the voltage limits of the analog tape recorder. This was done by subtracting a DC voltage from the input signal and then amplifying this signal. The units could also provide low-pass filtering at 18 db/octave. The filters were used to reduce aliasing noise from the tape recorder.

The output signal from the auxiliary units was recorded by a Lyric TR61 tape recorder with FM recording electronics. The recorder was provided with up to 8 channels of simultaneous recordings, which were not all used. It also had three recording speeds 0.6, 6.0 and 60 ips, where only the 60 ips speed was used. At this speed Lyrec quoted that the signal to noise ratio was greater than 40 db, and the measured frequency response was flat up to 20 kHz for all of the channels that were used.

2.3 THE COMPUTER FACILITIES

The laboratory computer faciltities comprised a PDP-11/34a central processor with 256 k bytes of main memory and a FP11-A floating point processor. This offers the user maximum memory and high speed floating point computations. Two RL01 random access, removable disk drives were used for secondary storage. One disk had to be resident to hold system utilities and source code as the operating system was overlayed (not all system tasks were in memory). The other disk was available for either source code or data. At the time the experiments were done there was only one video display terminal, a VT-100. This terminal served as the system console and for user access. A second terminal has been added to the system since then. Hard output was obtained through a LA-180, 180 character per second line printer. An HP 7475a, 6 pen plotter was used to draw either 8.5X11 or 11X17 graphs. For data acquisition the computer was equiped with a LPA-11K laboratory peripheral accelerator, which had a KW-11K dual programmable clock and an AD-11K 12 bit A/D converter. Data acquisition is discussed in detail in appendix C. A RSX-11M operating system was used along with a DEC FORTRAN compiler, and scientific and laboratory subroutine packages. Programs and programming details are described in the appendices.

TABLE 2.1

The location of the plugs in the diffuser.

	Plug number							
x (cm)	θ=0°	45°	90°	135°	180°	225°	270°	315°
4.62	65	66	67	68	69	70	71	72
15.59	57	58	59	60	61	62	63	64
19.58	49	50	51	52	53	54	55	56
23.57	41	42	43	44	45	46	47	48
31.55	33	34	35	36	37	38	39	40
39.53	25	26	27	28	29	30	31	32
50.71	17	18	19	20	21	22	23	24
55.69	9	10	11	12	13	14	15	16
69.46	1	2	3	4	5	6	7	8

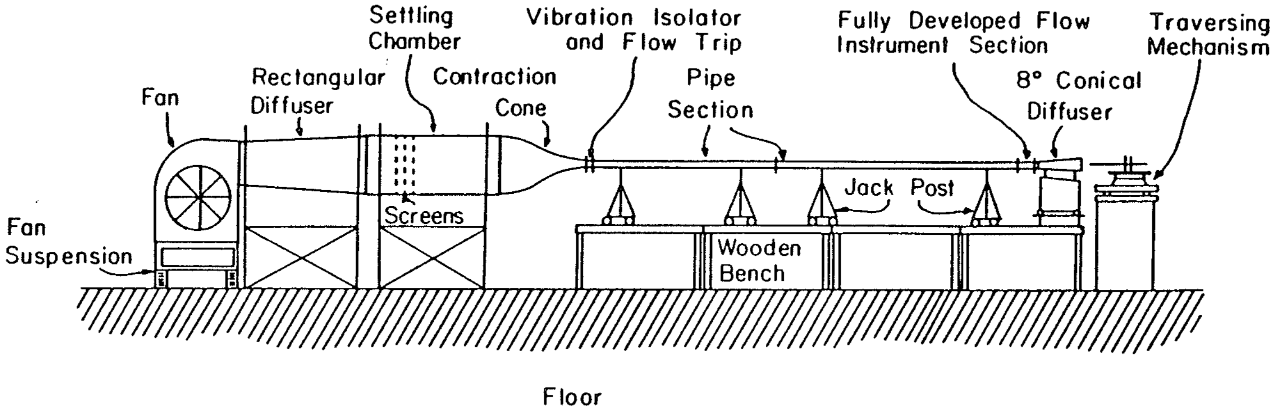
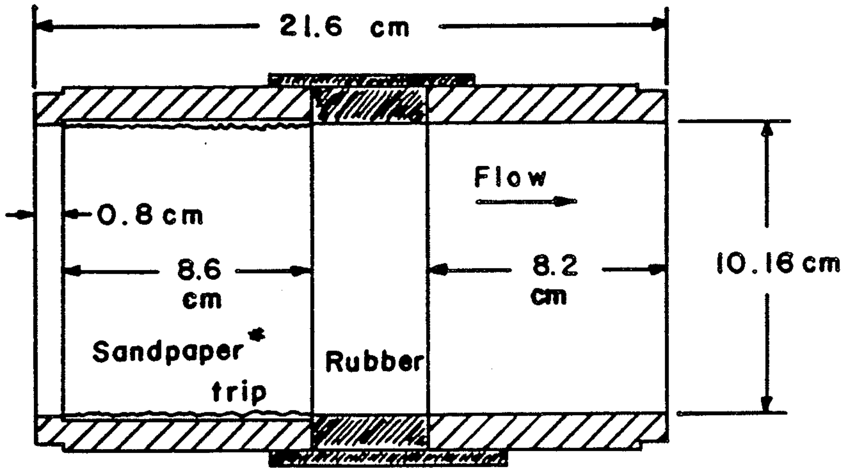


Figure 2.1: The general layout of the wind tunnel.

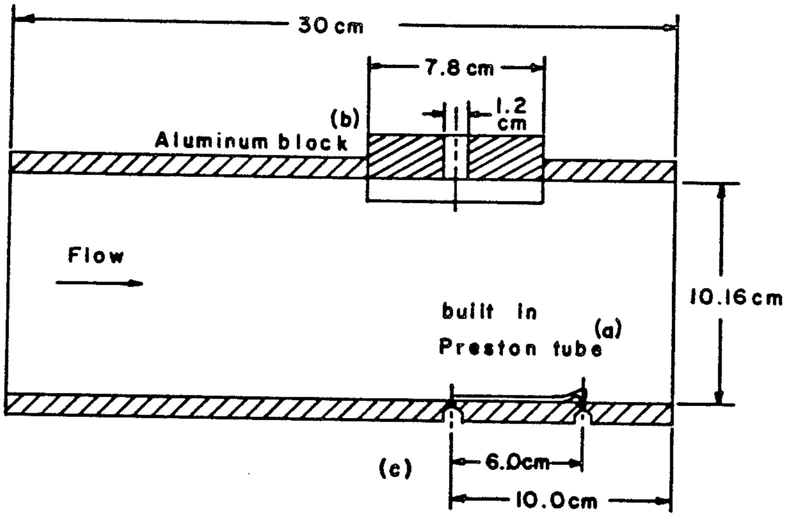


* Behr-Manning Division, Norton, Troy New York

*4 Grit Floor Sanding Combination

The pipe was made from steel hydraulictubing.

Figure 2.2: The combined vibration isolator and flow trip.



a) The Preston tube was made of 0.5 mm hypodermic tubing, epoxied in place.

b) The aluminum block was epoxied in place and was fit with a plastic plug which was used to hold the hot film shear stress probe.

c) Pressure taps were 1mm dia. (#60 drill) with 0.80 cm brass fittings epoxied on the outside of the pipe.

The pipe was made from steel hydraulic tubing.

Figure 2.3: The fully developed flow instrument section.

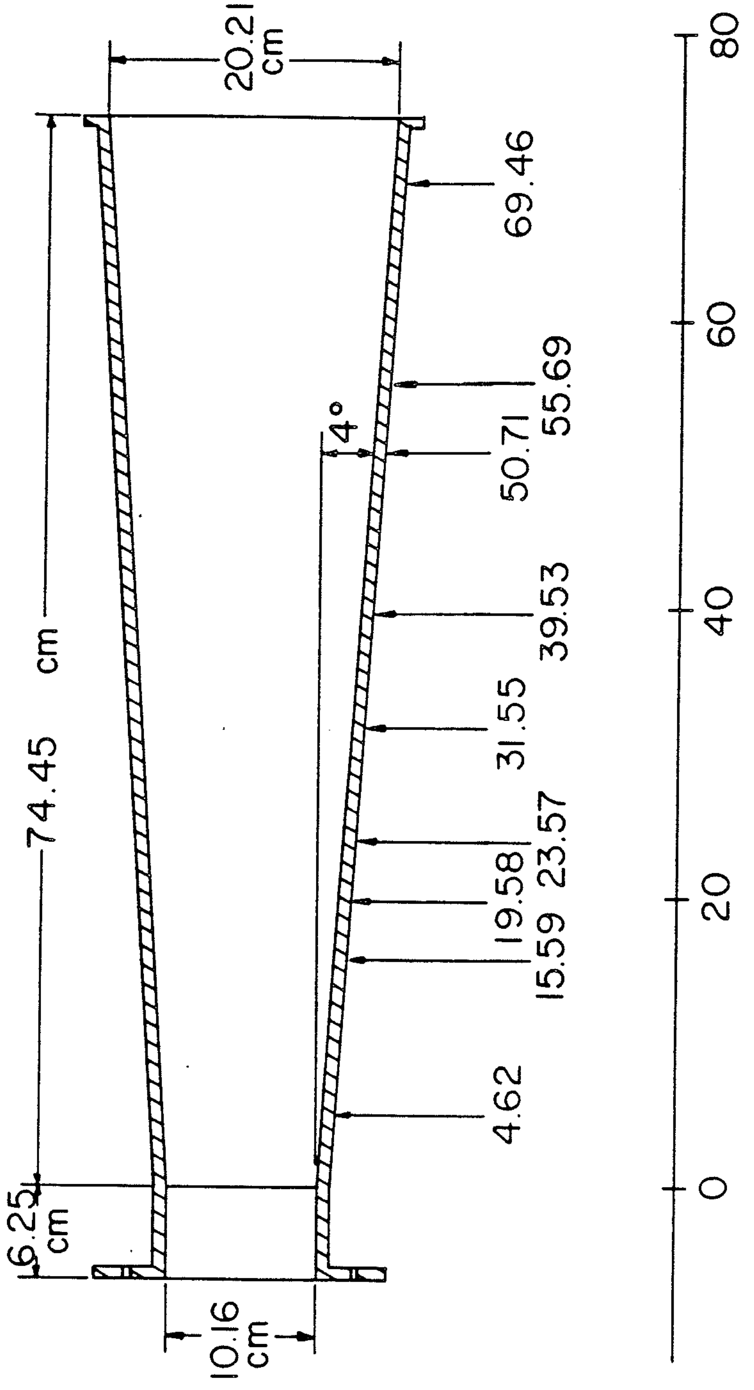


Figure 2.4: The eight degree conical diffuser. The arrows indicate the axial locations of the hot-film probe openings.

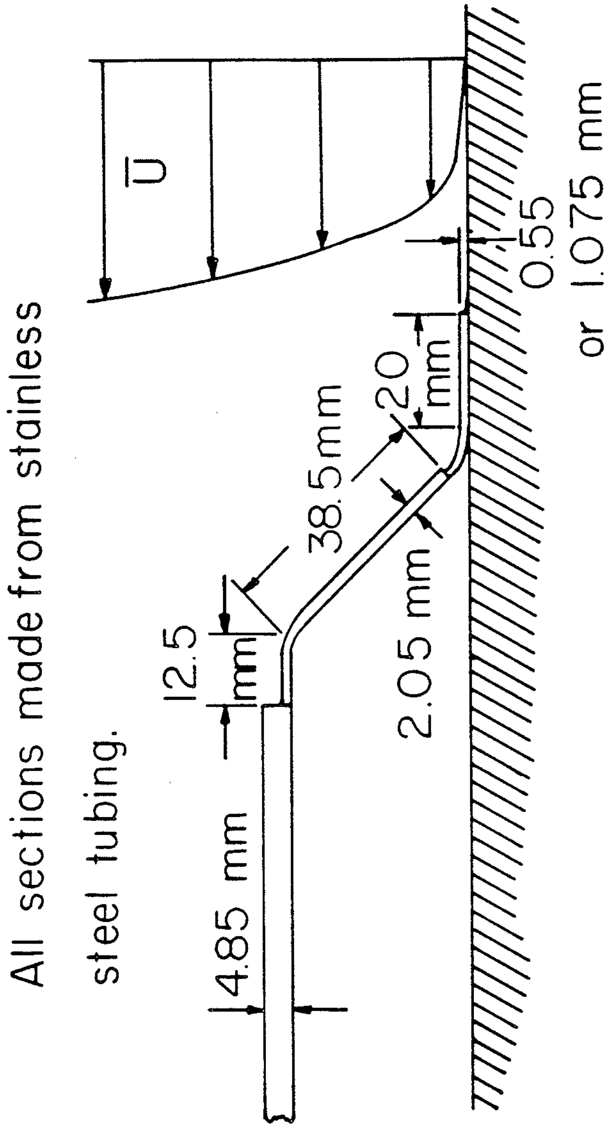


Figure 2.5: The Preston tube geometry.

Chapter III

Mean Wall Shear Stress Measurements

A compilation of the variation of the mean wall shear stresses, at various axial locations, as the wind tunnel control variable (the contraction cone pressure drop) changes are given in figure 3.1. The wall shear stress can be approximated by a linear function of the contraction cone pressure and as a linear function of the square of the Reynolds number. The Reynolds number is proportional to the square root of the contraction cone pressure drop.

As noted previously, two Preston tubes of different diameters were used. Both tubes indicated the same wall shear stress within experimental error throughout the diffuser. The measurements were also made on opposite sides of the diffuser and the wall shear stresses were found to be identical. Hence the flow was symmetric. Each combination of the tube diameter and side of the diffuser was measured twice. From these measurements it was found that the error in the indicated shear stress had a span of approximately ±15% of the mean of all of the measurements. This was attributed to the slight variations in geometry due to misalignment and other factors.

The published reports on Preston tube measurements, described in appendix A, do not outline some of the very serious sources of difficulty in using this method. For example, it was found that the indicated static pressure and hence Preston pressure could be adversely affected by having the Preston tube over the static pressure tap. Another problem that was encountered was that althought the percentage change in the length of the wind tunnel with temperature was small the large length of the tunnel made the position of the probe move around up to 1 mm. The probe position had to be continually adjusted to account for this. It is believed that the random positioning error caused the magnitude of the error to be as large as it was. Short of using built in Preston tubes, no solution to this problem can be seen at this time.

It was found that even in the worst case the correction to the inferred wall shear stress due to adverse pressure gradient was less than 5% of the wall shear stress. This correction was obtained from the results of Frei and Thomann [1980] as discussed in appendix A.

The lines in figure 3.1 are the average of the least square best fit lines for each experimental run. This data is summarized in table 3.1. These average lines can be taken as a fairly reasonable estimate of the variation of the wall shear stress with the contraction cone pressure.

The variation of the mean static pressure is shown in figure 3.2. It shows the initial very rapid increase in static pressure which decreases very rapidly to a nearly steady rate of increase. This is typical of this type of flow and is also found to be of a similar nature in the 6° conical diffuser of Pozzorini [1976].

Figure 3.3 shows the variation of the wall shear stress with axial position at a cone pressure of 1.0 in H_2O or a Reynolds number of 130,000, along with the estimated error bars. The wall shear stresses can be seen to fall off dramatically in the initial part of the diffuser and then decrease much slower. The calculated friction velocities are given in figure 3.4 and their behaviour is similar to that of the wall shear stresses.

A fixed Preston tube had been used in the pipe. The measured wall shear stresses still exhibited a large scatter as shown in figure 3.1. The mean trend did give values of wall shear stresses that corresponded to the values inferred from the Blasius resistance formula given by Schlichting [1979] for smooth pipes.

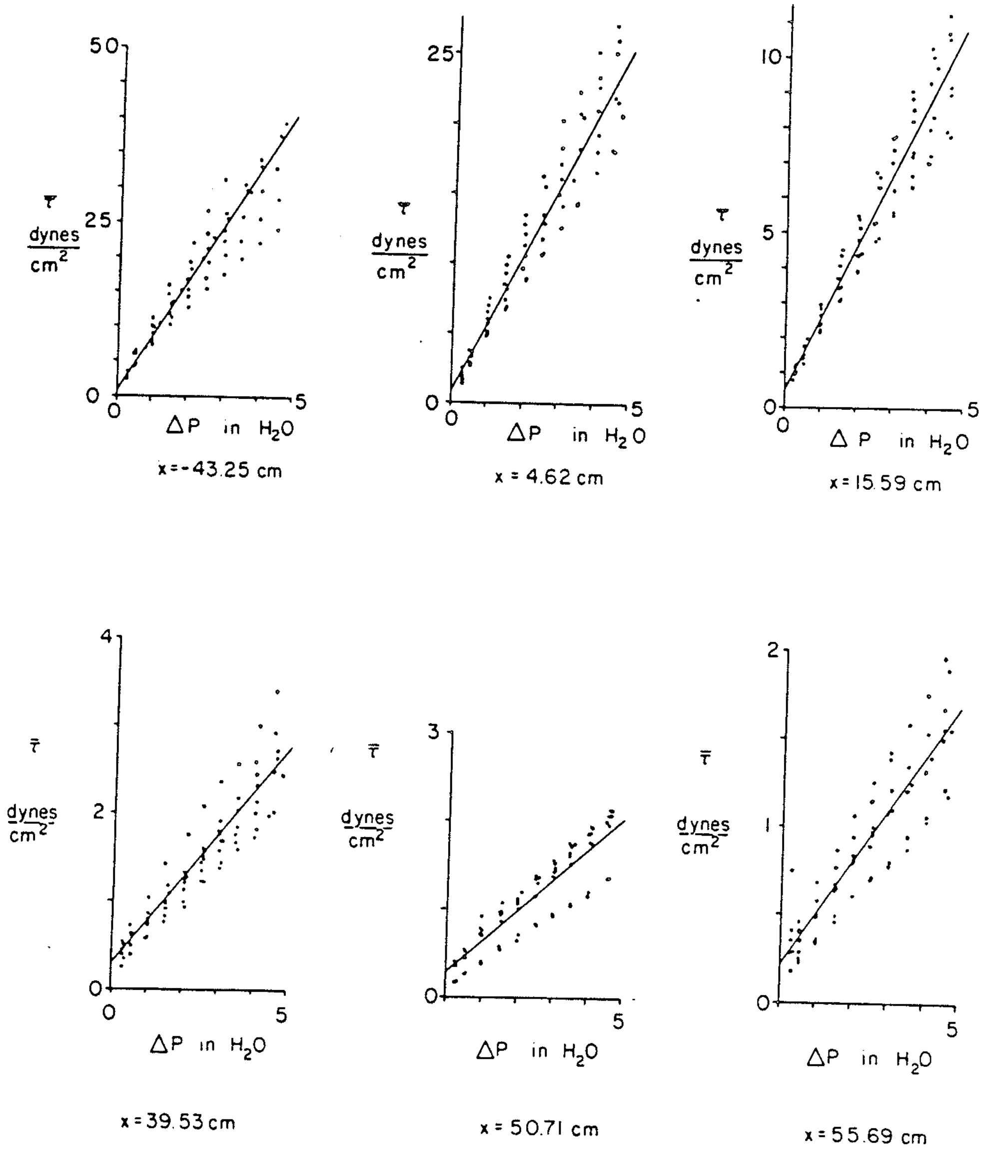
TABLE 3.1

The variation of τ with ΔP .

$$\bar{\tau} = A\Delta P + B$$

$\bar{\tau}$ - dynes/cm^2 , ΔP - in H_2O

x (cm)	A	SD(A)	B	SD(B)
-43.25	7.7415	1.6172	1.0285	0.4941
4.62	4.8597	0.7892	0.8162	0.1744
15.59	2.0626	0.4476	0.4476	0.1970
19.58	1.3273	0.2137	0.3172	0.1526
23.57	0.8867	0.1835	0.4241	0.1052
31.55	0.6757	0.1302	0.2881	0.0723
39.53	0.4972	0.0938	0.2912	0.1118
50.71	0.3511	0.0527	0.2834	0.1178
55.69	0.2926	0.0601	0.2182	0.0780
69.46	0.2001	0.0170	0.1824	0.0769



x=-43.25 cm

x = 4.62 cm

x = 15.59 cm

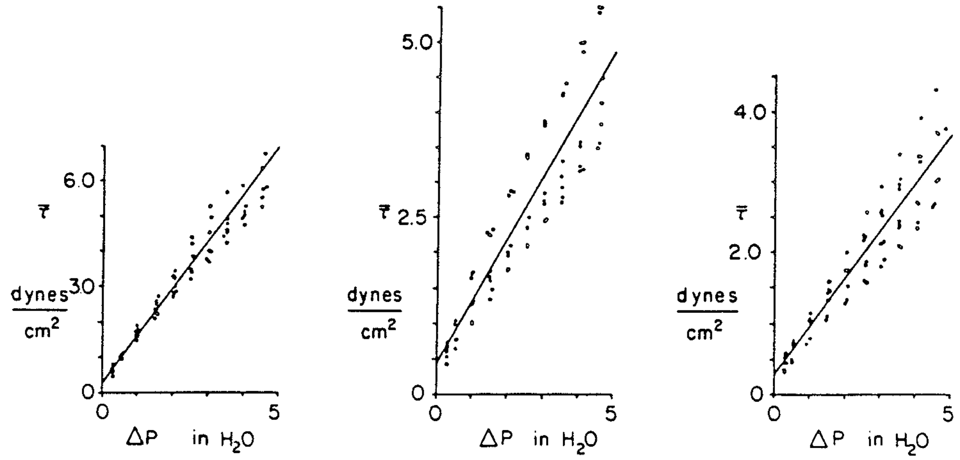
x = 39.53 cm

x = 50.71 cm

x = 55.69 cm

Figure 3.1: The variation of τ with ΔP .

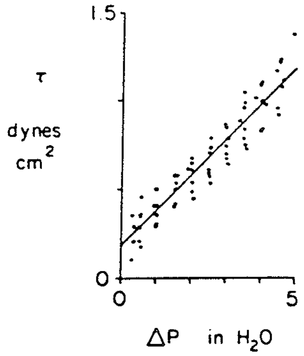
Continued on the next page.



x=19.58 cm

x = 23.57 cm

x=31.55 cm



x=69.46 cm

Figure 3.1 (cont.)

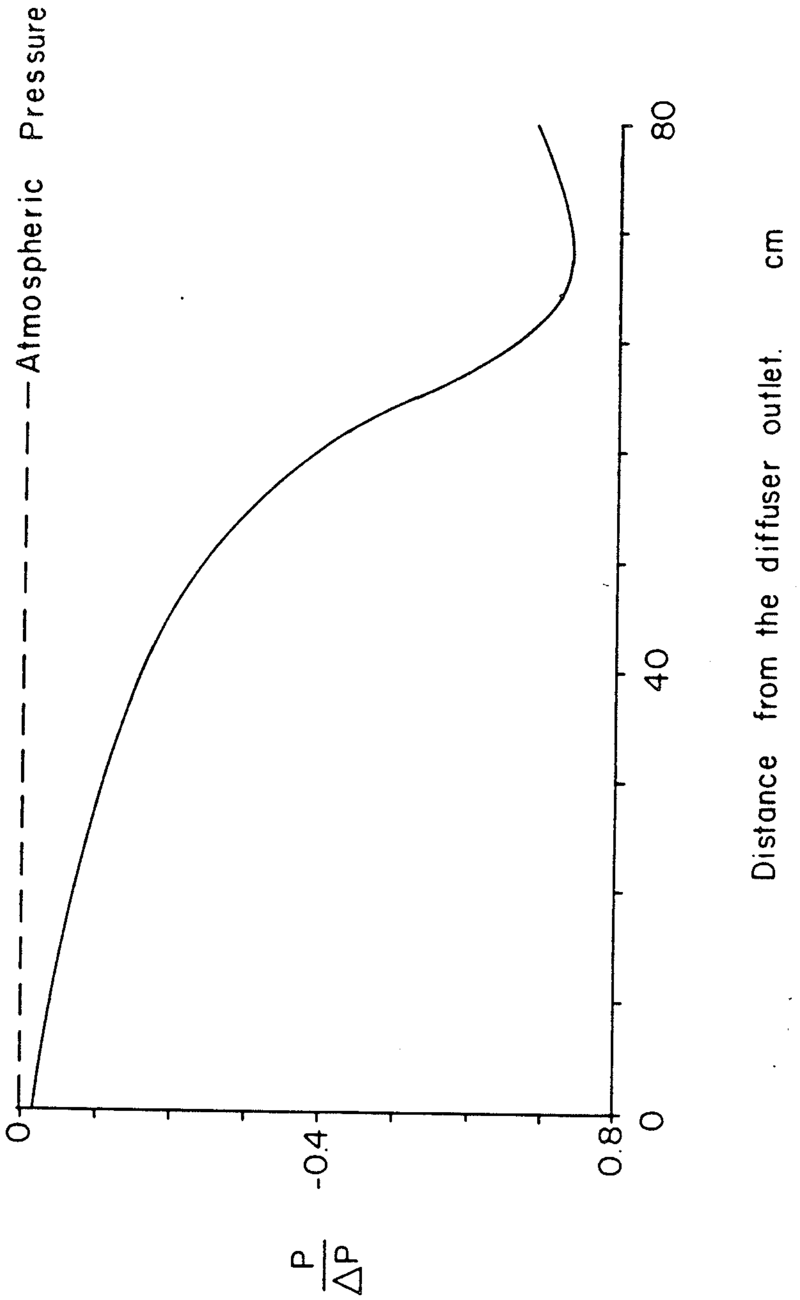


Figure 3.2: The axial distribution of the wall static pressure.

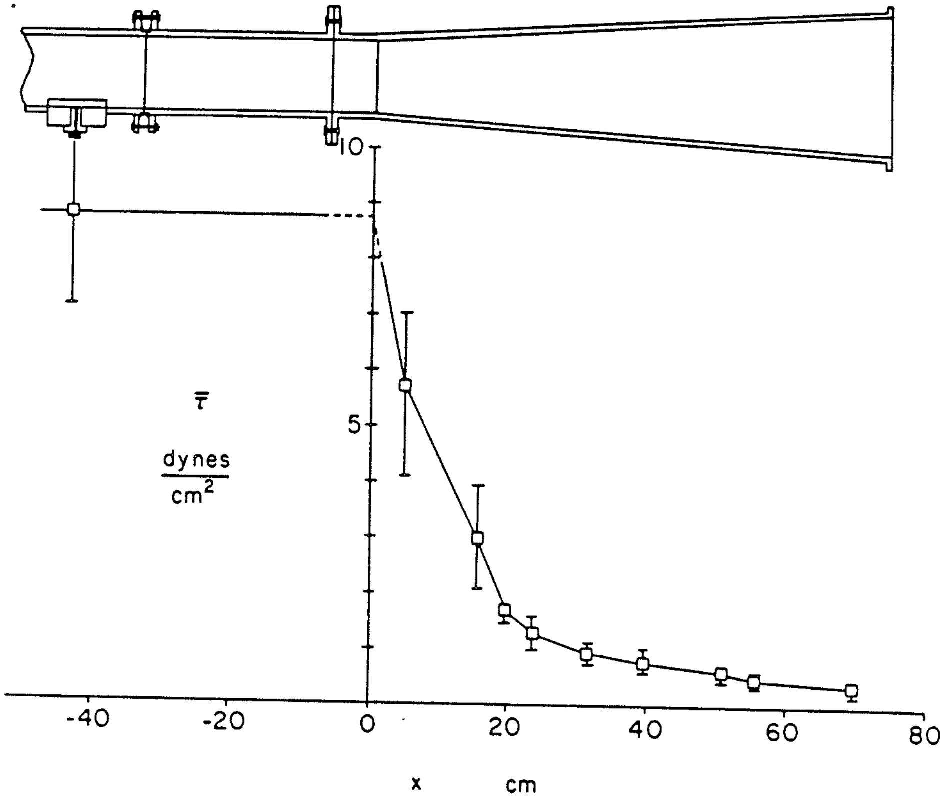


Figure 3.3: The axial distribution of $\bar{\tau}$.

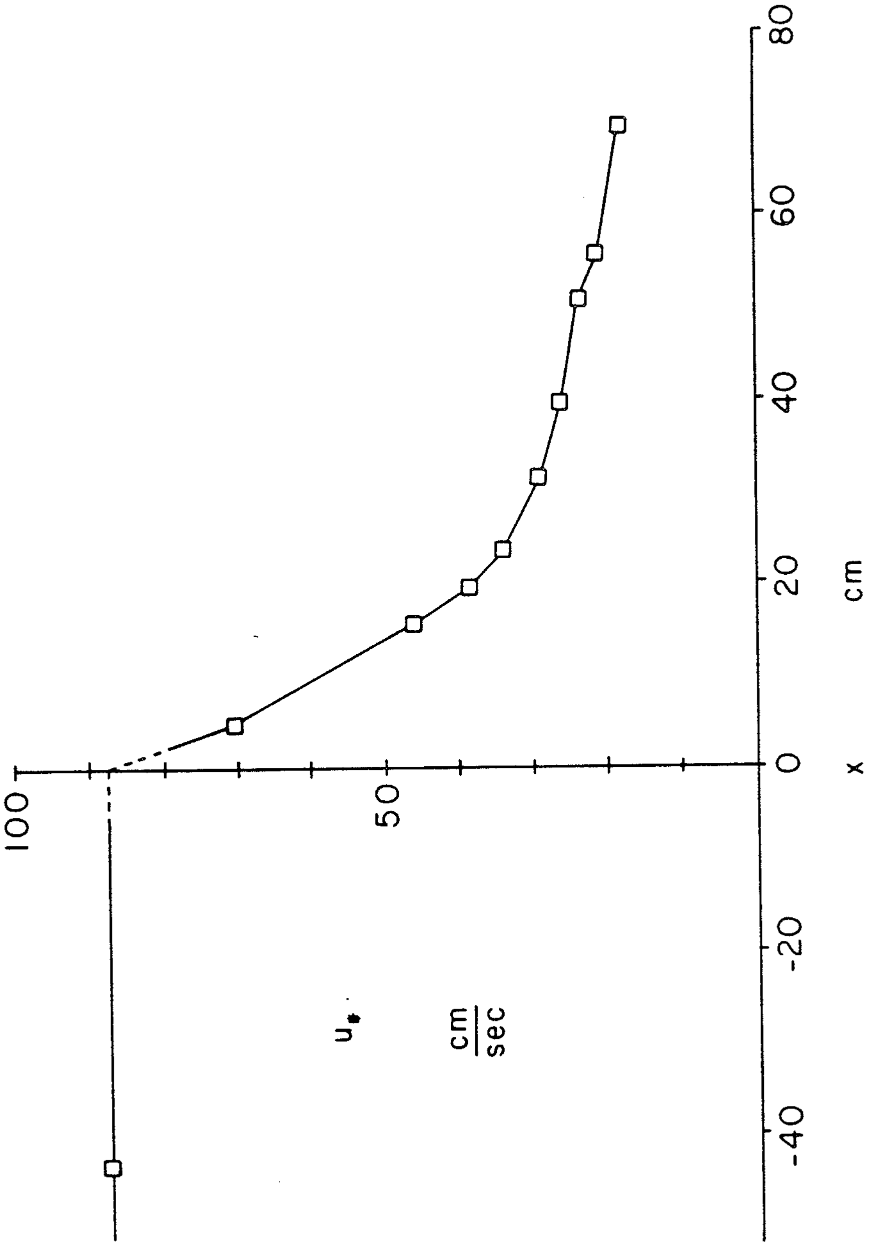


Figure 3.4: The axial distribution of u_*.

Chapter IV

Single Point Measurements

These measurements comprise all of the measurements of the fluctuating wall shear stress τ made at a single point. They are the RMS, the central moments up to 6th order, spectra, autocorrelation, the probability distribution and density functions and a decomposition based on Zaric's [1974,1975] conditional sampling scheme. Each of these will be discussed as an individual topic.

4.1 THE RMS OF τ

The measured axial distribution of τ' is shown in figure 4.1 for a Reynolds number of 130,000. In the immediate vicinity of the inlet the value of τ' is nearly constant for about 1/2 a diameter of the pipe (5 cm). The value of τ' drops very rapidly after this region and this rapid decrease reduces toward the outlet where τ' decreased slowly. Table 4.1 is a list of $\bar{\tau}$ and τ', where τ' is the average of many measurements. The error in this quantity is also given, and it can be seen to be very large, 10 to 20 percent. This large error can be directly attributed to the difficulty of estimating the sensitivity of the hot-film to the fluctuation in τ.

Figure 4.2 is a graph of $\tau'/\bar{\tau}$. This nondimensionalization changes the RMS shear stress picture considerably. Here the intensity shoots up very rapidly to a peak value of 0.55 at about 1/3 of the diffuser's length. Over the next 1/3 of the length the intensity decreases to about 0.45 and in the final 1/3 the value of this intensity remains fairly constant at this value.

Of course, with the magnitude of the error in these measurements it would be dangerous to try and draw too detailed conclusions from this data. But, the general picture appears to support the conclusion that the sublayer is 'passive' and only responds to external stimuli. The character of the $\tau'/\bar{\tau}$ distribution is such that at a point the intensity is determined by the fluctuations that are generated upstream from that point. In this way τ' would respond to a change in pressure gradient after or downstream from that change. Hence local changes that affect the turbulence structure will manifest themselves downstream from that point.

4.2 THE CENTRAL MOMENTS OF τ

The distribution of the skewness of the wall shear stresses $S(\tau)$ is shown in figure 4.3. The value increases from the value in the pipe gradually throughout the entire diffuser. It is very dificult to say whether this change is a real change as the value of $S(\tau)$ in the pipe is lower than most measurements that have been reported in the past. Py and Duhamel [1972] reported a value of approximately 0.55 for $S(\tau)$ in a pipe and Eckelmann [1974] gave a value of 0.75 in a very thick viscous sublayer in a channel flow.

The flatness factor of the wall shear stress $F(\tau)$ is given in figure 4.4. The value of $F(\tau)$ is very nearly 3.0 in the pipe and remains near this value for roughly the first 2/3 of the diffuser. After this $F(\tau)$ increases slowly to a value of roughly 3.7 near the outlet.

The super-skewness $K_5(\tau)$ and the super-flatness $K_6(\tau)$ are shown in figures 4.5 and 4.6 respectively. $K_5(\tau)$ has a value of 2 in the pipe and uniformly increases in the diffuser to a value of ~8 at the outlet of the diffuser. $K_6(\tau)$ has a large value of 14 in the pipe and this value increases to about 30 at the outlet.

It is apparent from these last two figures that the error increases as the order of the moment increases. This is due to two causes. First as the order of the moment increases the weighting of the extreme events increases. Because of this the moments will be more strongly contaminated by nonlinearity in the calibration. Secondly these extreme events have a lower probability of occuring as their magnitude increases. Hence to reduce random sample fluctuation error a larger sample is required in order to estimate a higher order moment. This is rarely done.

Using the value of $y_+=5$ for the extent of the sublayer thickness gives a viscous sublayer thickness of 0.09 mm in the pipe and 0.54 mm near the outlet of the diffuser, a 6 fold increase. The increase in the moments might very well be due to the improved response of the hot-film as one moves downstream rather than a change in the character of the sublayer.

Another source of error, a systematic error, is that due to the nonlinear response of the hot-film. The slope of the calibration curve decreases as the mean value or operating point increases. Practically, using electronic linearizers there is a limit to the range over which the signal can be calibrated. Thus if a fluctuation with a given operating point had its operating point reduced the odd moments would increase in magnitude and if its operating point increased they would decrease.

In the light of these severe limitations all that can be said about the central moments is that they increase slightly from the diffuser inlet to the outlet, and that the odd moments are positive.

4.3 THE SPECTRA OF τ

The dimensional spectra of τ are shown in figures 4.7 through 4.16 inclusive. The reason for showing each one individually is for graphical clarity. A combined figure is a confusing jumble of lines.

Just as in the spectra of u in most turbulent flows, the energy containing range of the spectra of τ is at very low frequencies. In the pipe where the τ spectrum has the largest high frequency extent most of the energy is below approximately 100 Hz. Towards the outlet of the diffuser this region falls to within even lower frequency limits. The spectra all showed a range of power law decrease at the high frequency end of the energy containing range of frequencies. This power law decrease appears to be a f^{-1} decay relationship. In the pipe it extends over roughly a 2 decade frequency band which reduces to about a 1 decade band at x=69.46 cm. Following this f^{-1} region the rate of decay of the spectra increases very rapidly. Table 4.2 is a list of the outer scaling variables for each point as well as the upper frequency limits of the spectra. The upper frequency limits, fc, had been determined from pilot measurements using a very high frequency sampling rate. The fc were set at the high frequency spectral minima of the pilot spectra. At this point noise overwhelms the spectra. The fc were 10 kHz in the pipe and the initial portion of the diffuser, after this fc decreased very rapidly up to about 1/2 the diffuser's length. In the last half of the diffuser fc decreased at a much lower rate, and was 2 kHz at the last measurement point x=69.49 cm.

Just before fc, the spectra showed a narrow range of another power law decay. The bandwidth was in the order of 1/2 a decade and the rate of decrease was greater than $f^{-5/3}$

When the τ spectra were nondimensionalized by the outer time scale and τ' they were found to collapse to a universal form. This is shown in figure 4.17.

4.4 THE AUTOCORRELATION OF τ

The autocorrelation of τ are shown in figures 4.18 through 4.27 inclusive. They are plotted in two ways: linear and semilogarithmic (logΔt). The linear plots emphasize the long time delay characteristics and the semilog plots show more detail at small time delays.

The maximum time delay was 1.28 sec at x=69.46 cm and 0.256 sec in the pipe. Even with these long delays the autocorrelation function did not go to zero. From the linear plot it appears that the function has attained an asymptotic value. This is not necessarily true, the semilog plots show that the autocorrelation would require another order of magnitude in the time delay to become zero. This indicates that the long time scale, where R (Δt) goes to zero, is in the order of 2.5 sec to 10 sec. This indicates that special handling procedures are required to measure these values. These very low frequencies had not been anticipated, and hence were not dealt with. The signal would have to be sampled at a fairly low rate for very long periods for time to accommodate sub-Hertz frequencies. The autocorrelation functions had been integrated to obtain the integral time scales at each point. They are given in table 4.3 along with the outer time scale D_0/Um and a 'corrected' integral time scale where the minimum correlation times the maximum time delay has been subtracted. As can be seen the outer time scale and the corrected integral time scale are nearly equal.

One of the interesting points to note from the semilog plots, is that R (Δt) is nearly a constant value of 1.0 at small time delays. This minimum time ranges from 0.05 msec in the pipe to 0.5 msec at x=69.46 cm in the diffuser.

4.5 THE PROBABILITY DENSITY FUNCTIONS OF τ

The probability distributions and the probability density functions of τ for all of the axial stations are shown in figures 4.28 through 4.37 inclusive. The probability density functions are bounded to within -3 standard deviations to 5 standard deviations. They also display positive skewness throughout, which is displayed most prominantly in the plots of log(PDF(τ)) against τ/τ'. The peak value of the Gaussian distribution is nearly 0.4, and this is approximately the same as the peak value of PDF(τ) as shown here.

The PDF(τ) have all been plotted on one linear graph in figure 4.38. From this figure it is evident that, within experimental error, the PDF(τ) are the same as the PDF(τ) in the pipe throughout the diffuser.

4.6 THE ZARIC DECOMPOSITION OF τ

The Zaric decompositions of the probability density functions are shown in figures 4.39 through 4.48 inclusive. In all cases, except for the anomalous result at x=50.71 cm, the ejection event is more probable that the sweep event. Integral measures of the decomposition are given in table 4.4. The non-ejection/non-sweep events are the dominant events however. They occur roughly 75% of the time, the ejection events occur roughly 15% of the time with the sweep events filling in the remainder. These intermittencies for the diffuser are shown in figure 4.49. From this figure it can be seen that the relative amount of time each event occurs is nearly constant and hence independent of the pressure gradient.

The relative contribution to the mean square of τ from each phase or event is given in table 4.5 and plotted in figure 4.50. In this figure it is obvious that the sweep events contribute more to the mean square of the skin friction than the ejection events even though the relative occurance of ejection is greater. The ejection events contribute to 25% of the mean square of τ, sweep events 35% and the non-ejection/non-sweep events 40%.

Figures 4.51 and 4.52 are plots of the skewness and flatness factors of each event, respectively. The relative uniformity of integral measures of the decompositions throughout the diffuser indicate that the structure near the wall is not influenced greatly by the adverse pressure gradient. Hence the picture that emerges is that although the magnitude or RMS of τ and the outer time scale are affected by the pressure gradient the local structure is not.

TABLE 4.1

The τ' and $\bar{\tau}$ data in the 8° conical diffuser.

x (cm)	$\bar{\tau}$ (dynes/cm^2)	τ' (dynes/cm^2)	sample size	SD(τ') %
-43.25	8.7700	2.1226	5	2.1
4.62	5.6789	2.2772	29	20.7
15.59	2.5102	1.3856	31	28.3
19.58	1.6345	0.8990	31	11.4
23.57	1.3108	0.6279	31	10.9
31.55	0.9638	0.5330	30	19.2
39.53	0.7884	0.3769	29	16.5
50.71	0.6345	0.2697	31	18.4
55.69	0.5108	0.2309	30	15.0
69.46	0.3825	0.1706	37	14.6

TABLE 4.2

The fc and outer scaling parameters of the spectra.

x (cm)	fc (kHz)	D_0 (cm)	Um (m/sec)	T_0 (msec)
-43.25	10	10.16	19.32	5.26
4.62	10	10.81	17.08	6.33
15.59	7	12.32	13.10	9.42
19.58	4	12.90	11.99	10.72
23.57	3.5	13.46	11.01	12.23
31.55	3.25	14.57	9.39	15.52
39.53	3	15.96	8.10	19.37
50.71	2.75	17.25	6.71	25.75
55.69	2.5	17.95	6.19	29.00
69.49	2	19.87	5.05	39.35

TABLE 4.3

The integral time scales from the autocorrelation.

x (cm)	I (msec)	Ic (msec)	T_0 (msec)	Ic/T_0
-43.25	24.74	8.46	5.26	1.61
4.62	18.53	6.62	6.33	1.05
15.59	12.84	12.94	9.42	1.37
19.58	39.29	20.09	10.72	1.87
23.57	61.75	25.95	12.23	2.12
31.55	84.67	45.57	15.52	2.94
39.53	74.00	37.48	19.37	1.93
50.71	94.98	47.91	25.75	1.86
55.69	88.52	48.43	29.00	1.67
69.49	143.61	60.79	39.35	1.54

TABLE 4.4

The Zaric decomposition data.

Total signal

x (cm)	mean	RMS	S	F
4.62	-0.018	1.001	0.760	3.836
15.59	-0.019	0.999	0.497	3.354
19.58	-0.014	0.998	0.298	3.116
23.57	-0.011	1.010	0.378	3.040
31.44	-0.018	0.993	0.474	3.223
39.53	-0.020	1.001	0.443	3.062
50.71	-0.014	0.996	0.758	4.010
55.69	-0.019	1.003	0.801	3.922
69.46	-0.011	1.011	0.660	3.485

Ejection events

x (cm)	mean	RMS	S	F	γ
-43.25	-0.996	1.106	-1.741	7.054	0.2584
4.62	-0.843	1.179	-0.922	12.147	0.1666
15.59	-0.989	1.195	-1.777	9.231	0.1660
19.58	-1.017	1.157	-2.099	9.014	0.1860
23.57	-1.012	1.154	-2.138	9.262	0.1654
31.55	-0.985	1.118	-2.042	9.496	0.1629
39.53	-1.035	1.115	-1.916	7.887	0.1806
50.71	-1.083	1.475	-2.849	99.480	0.0266
55.69	-0.999	1.112	-1.967	10.258	0.1454
69.46	-1.072	1.144	-2.338	9.562	0.1396

Continued on the next page.

Table 4.4 continued

Non-ejection/non-sweep events

x (cm)	mean	RMS	S	F	γ
-43.25	0.168	0.740	0.354	3.975	0.6685
4.62	-0.094	0.691	0.063	3.794	0.7241
15.59	-0.054	0.713	-0.085	3.858	0.7235
19.58	0.029	0.752	-0.111	3.938	0.7223
23.57	-0.021	0.769	-0.111	3.740	0.7337
31.55	-0.029	0.758	-0.090	3.663	0.7399
39.53	0.000	0.752	-0.108	3.628	0.7277
50.71	-0.177	0.776	0.127	2.932	0.8824
55.69	-0.060	0.739	0.171	3.547	0.7606
69.46	-0.045	0.760	0.129	3.437	0.7615

Sweep events

x (cm)	mean	RMS	S	F	γ
-43.25	1.728	1.775	3.958	18.832	0.0731
4.62	1.745	1.827	3.429	15.082	0.1094
15.59	1.666	1.731	3.351	14.600	0.1105
19.58	1.684	1.743	3.638	16.573	0.0916
23.58	1.711	1.750	3.384	14.678	0.1009
31.55	1.687	1.770	3.569	16.046	0.0972
39.53	1.801	1.820	3.491	15.376	0.0922
50.71	1.882	1.960	3.739	17.827	0.0910
55.69	1.828	1.952	3.752	17.829	0.0941
69.46	1.745	1.869	3.630	16.445	0.0989

TABLE 4.5

The contribution to the mean square by each event.

x (cm)	ejection %	non-ejection/non-sweep %	sweep %
-43.25	34.7	40.1	25.2
4.62	24.6	36.7	38.7
15.59	25.3	39.3	35.4
19.58	26.6	43.6	29.7
23.57	22.8	45.1	32.1
31.55	21.9	45.5	32.7
39.53	23.9	43.8	32.4
50.71	5.3	62.7	32.0
55.69	17.3	44.4	38.4
69.49	18.9	45.5	35.6

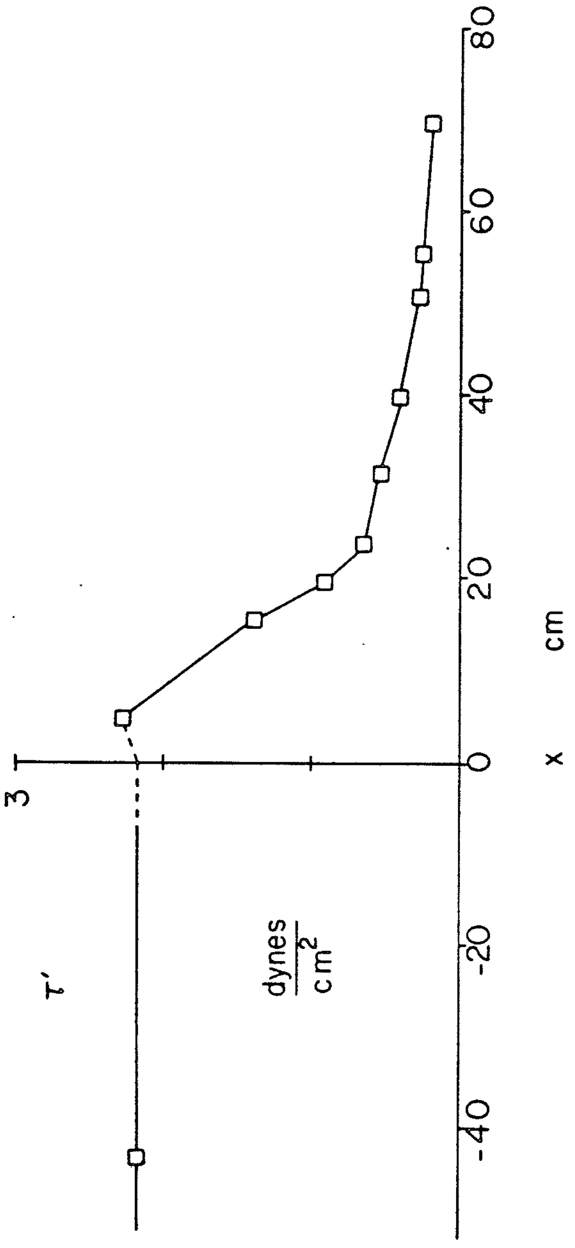


Figure 4.1: The τ' distribution in the 8° conical diffuser.

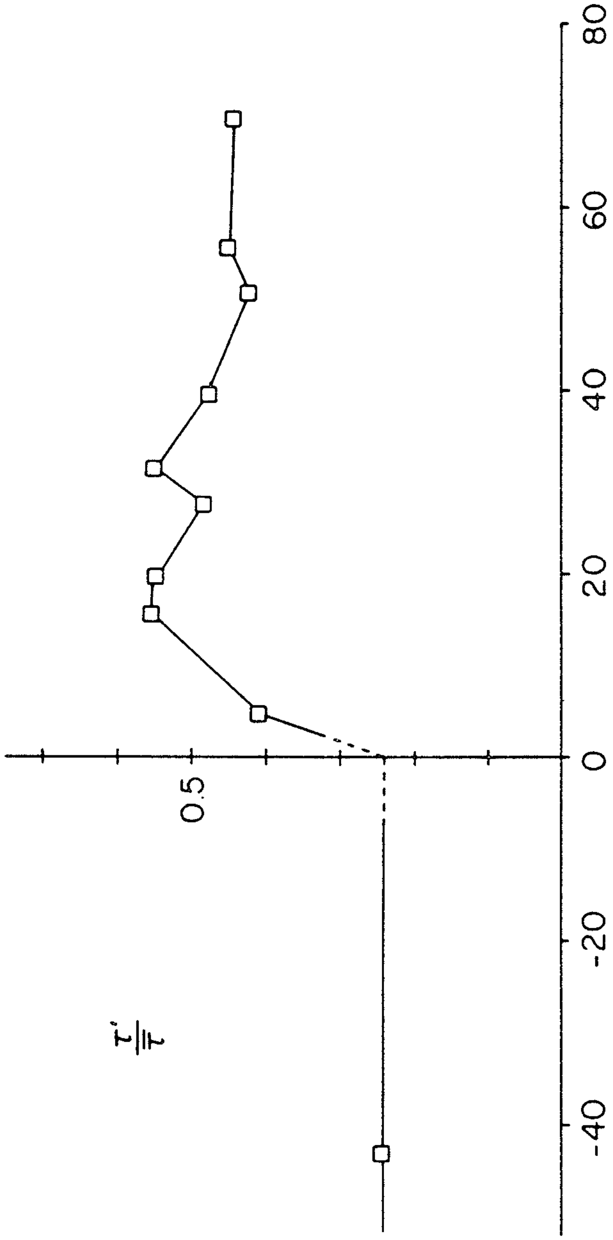


Figure 4.2: The $\tau'/\bar{\tau}$ distribution in the 8° conical diffuser.

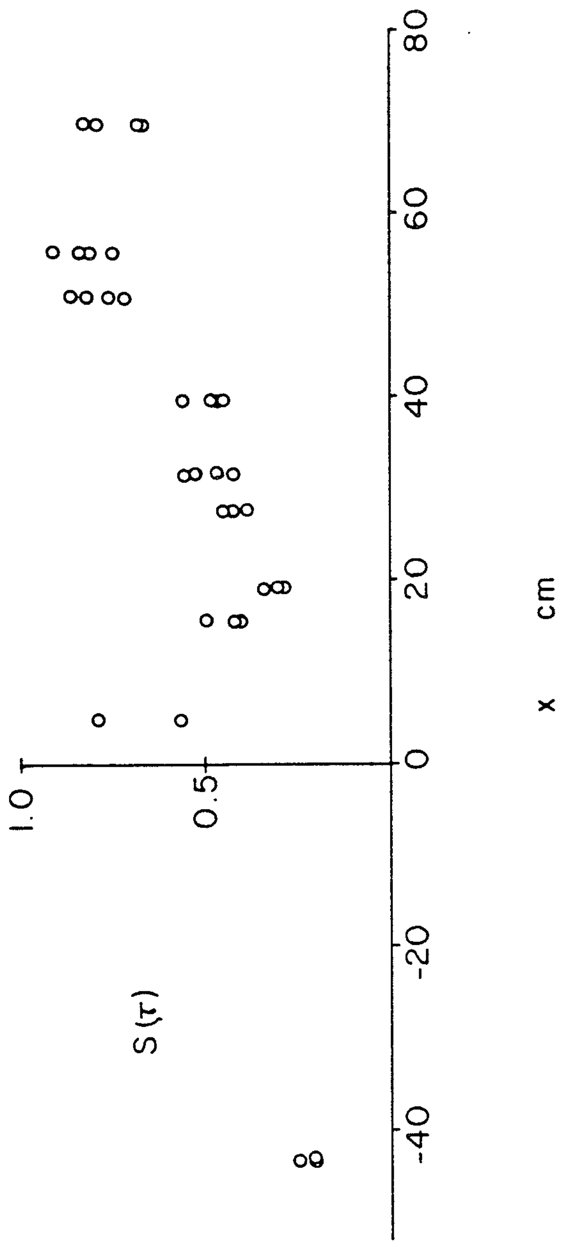


Figure 4.3: The S(τ) distribution in the 8° conical diffuser.

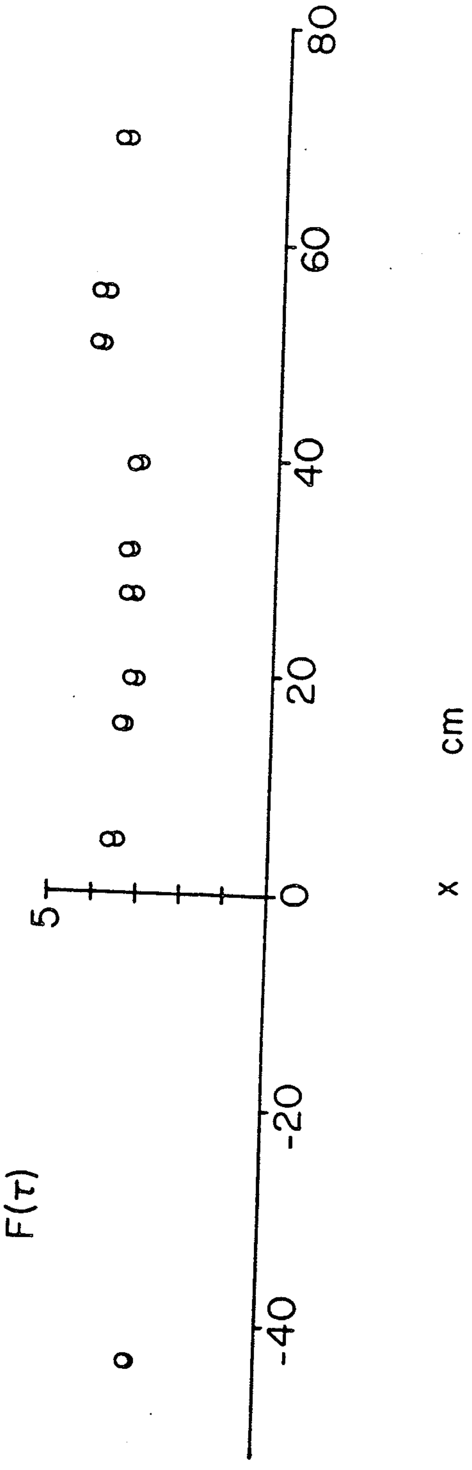


Figure 4.4: The F(τ) distribution in the 8° conical diffuser.

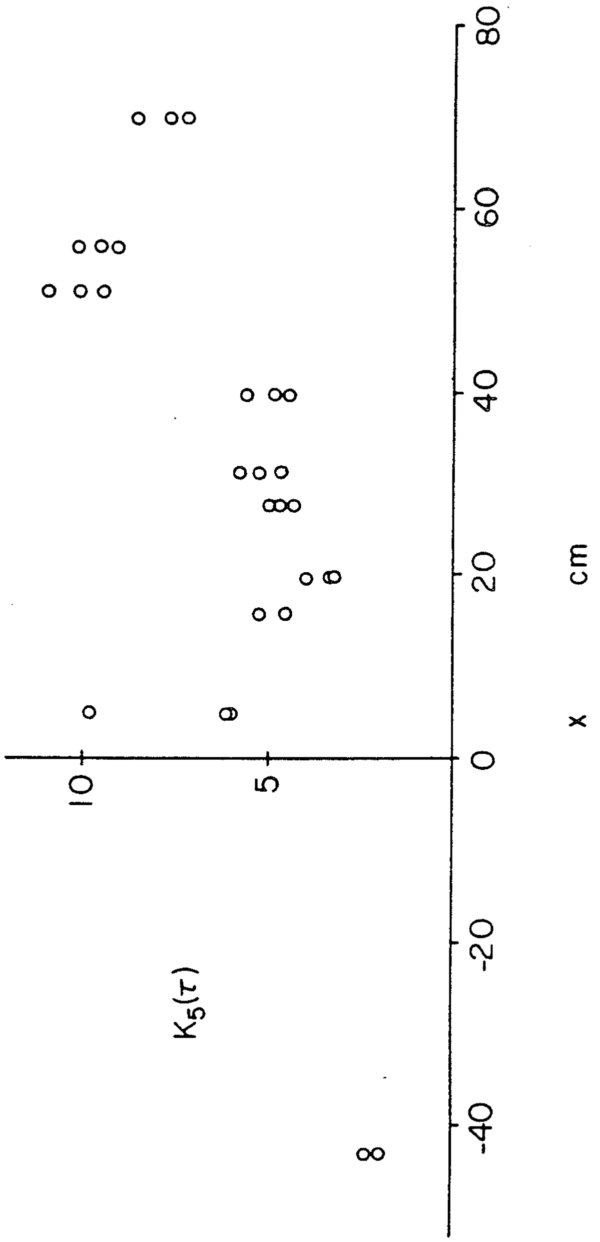


Figure 4.5: The $K_5(\tau)$ distribution in the 8° diffuser.

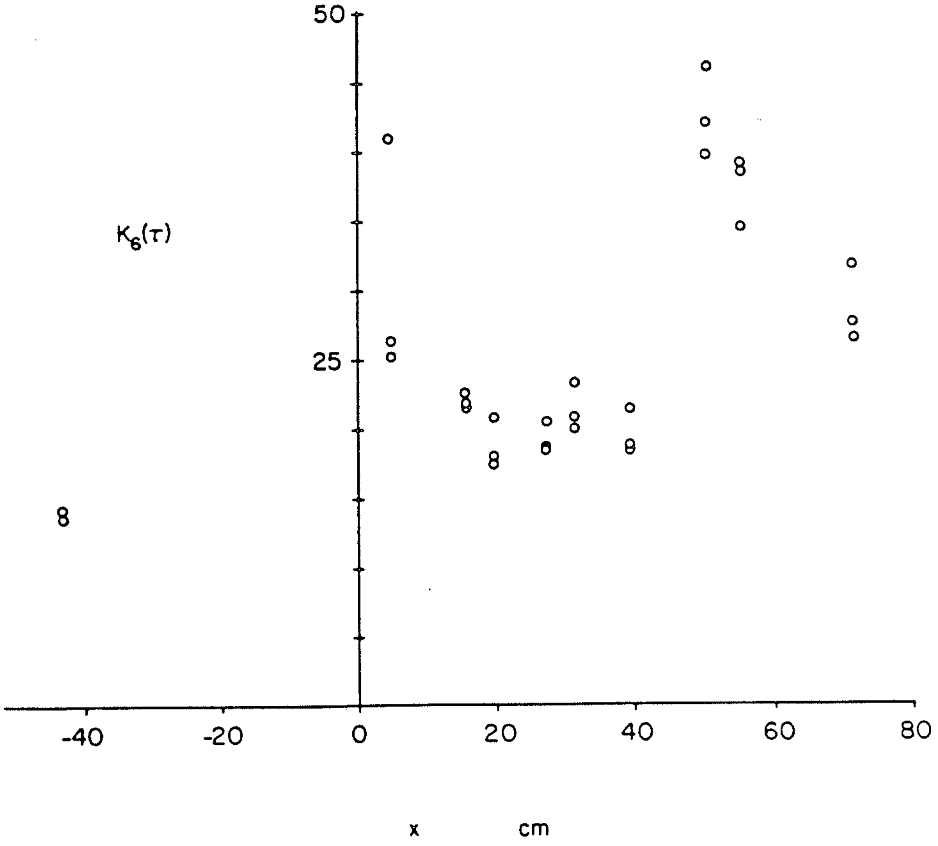


Figure 4.6: The $K_6(\tau)$ distribution in the 8° diffuser.

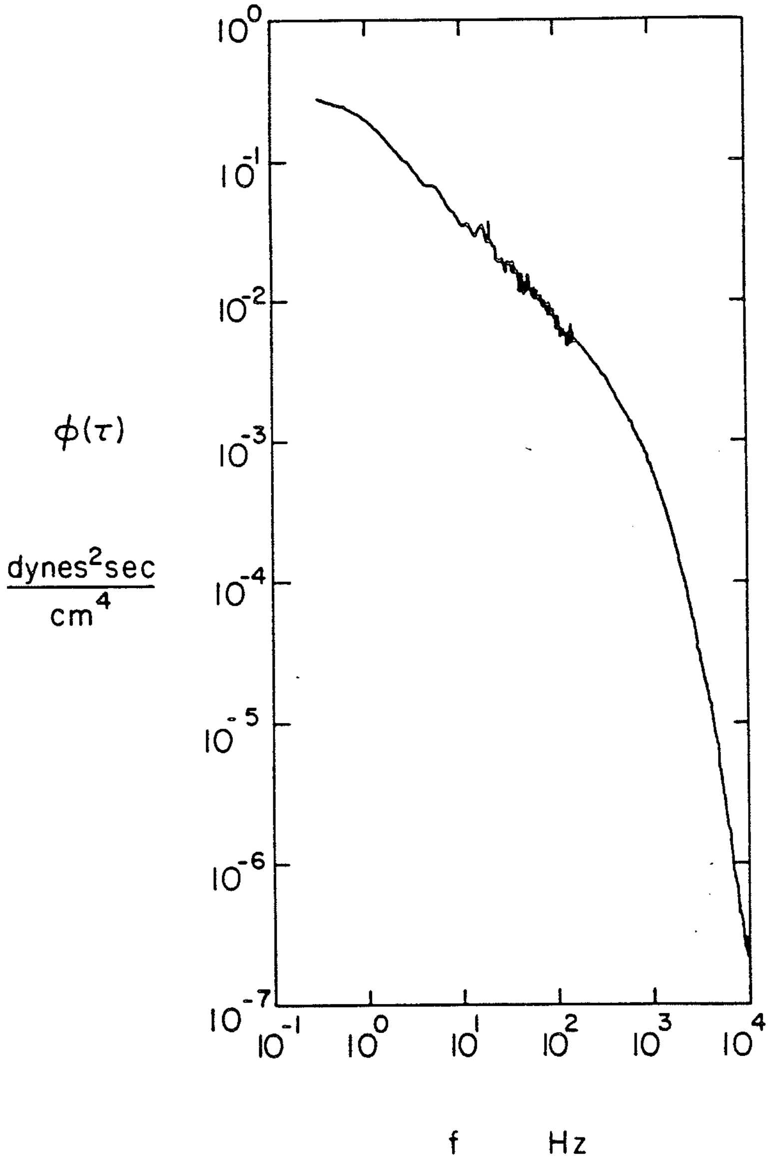


Figure 4.7: The τ spectrum in the pipe.

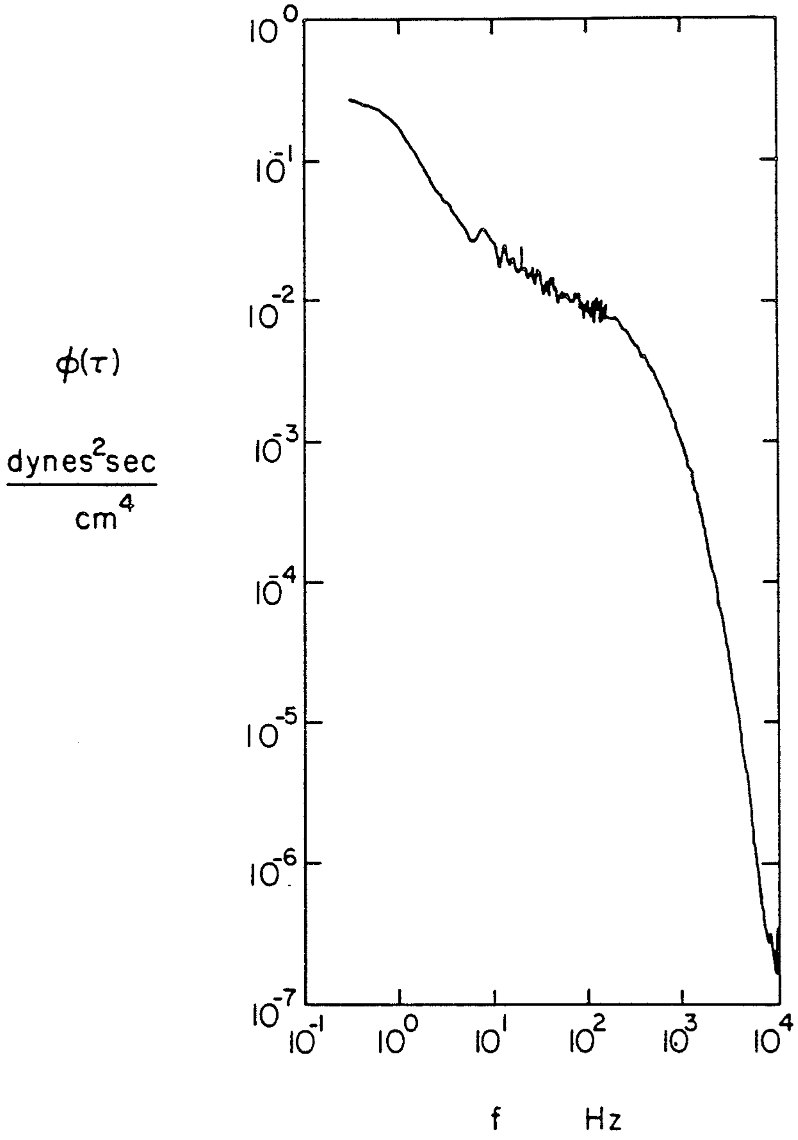


Figure 4.8: The τ spectrum at x=4.62 cm.

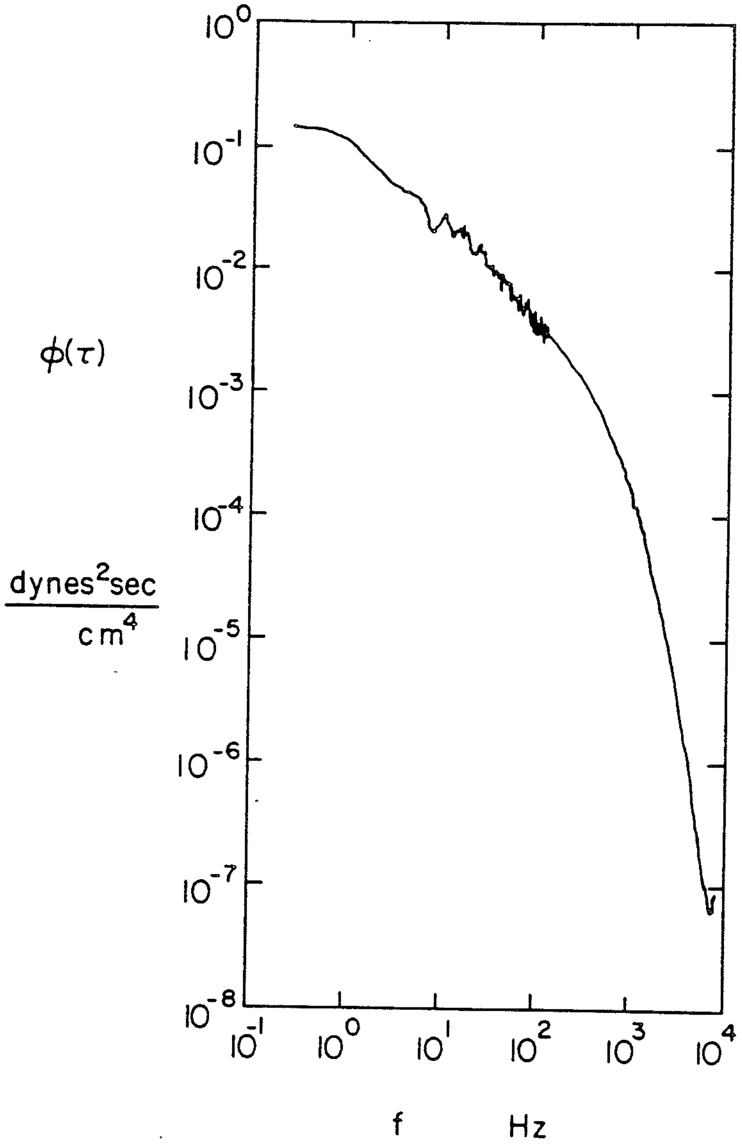


Figure 4.9: The τ spectrum at x=15.59 cm.

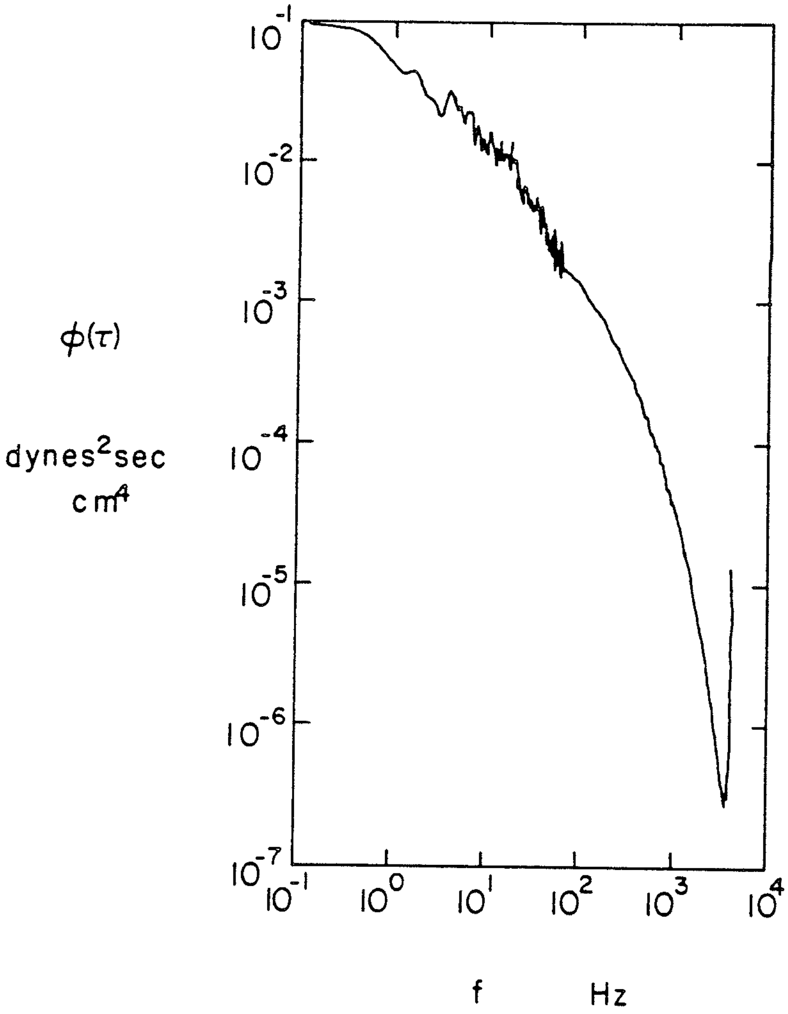


Figure 4.10: The τ spectrum at x=19.59 cm.

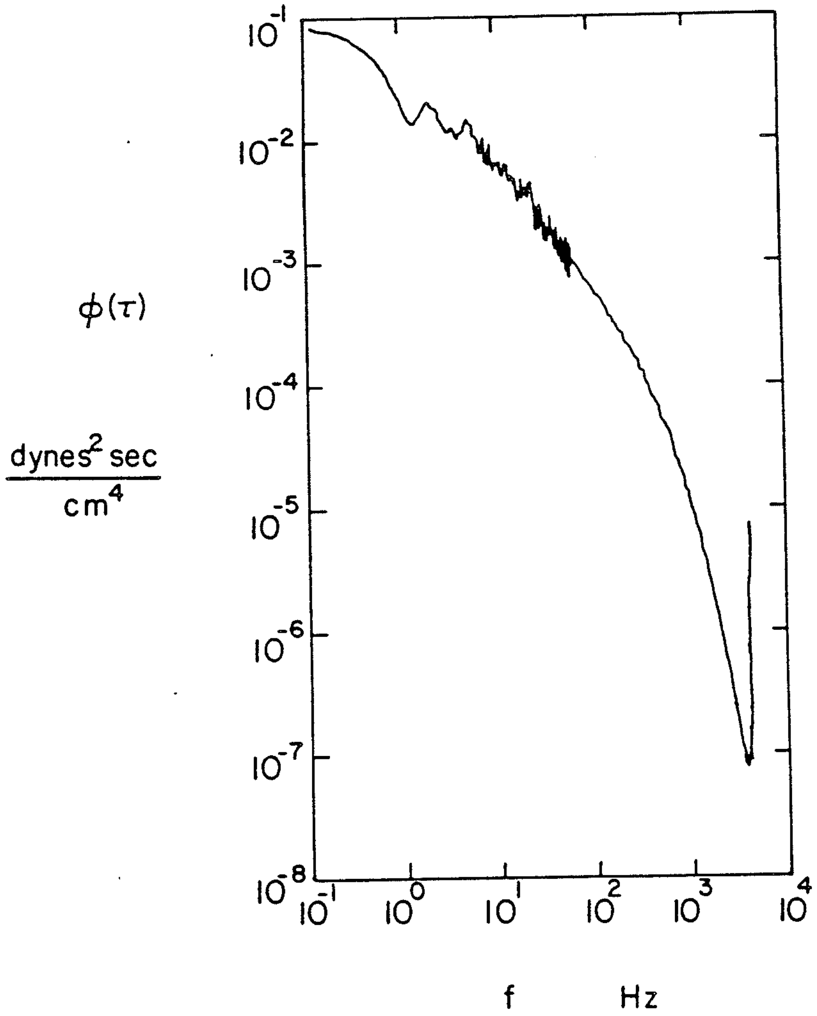


Figure 4.11: The τ spectrum at x=23.57 cm.

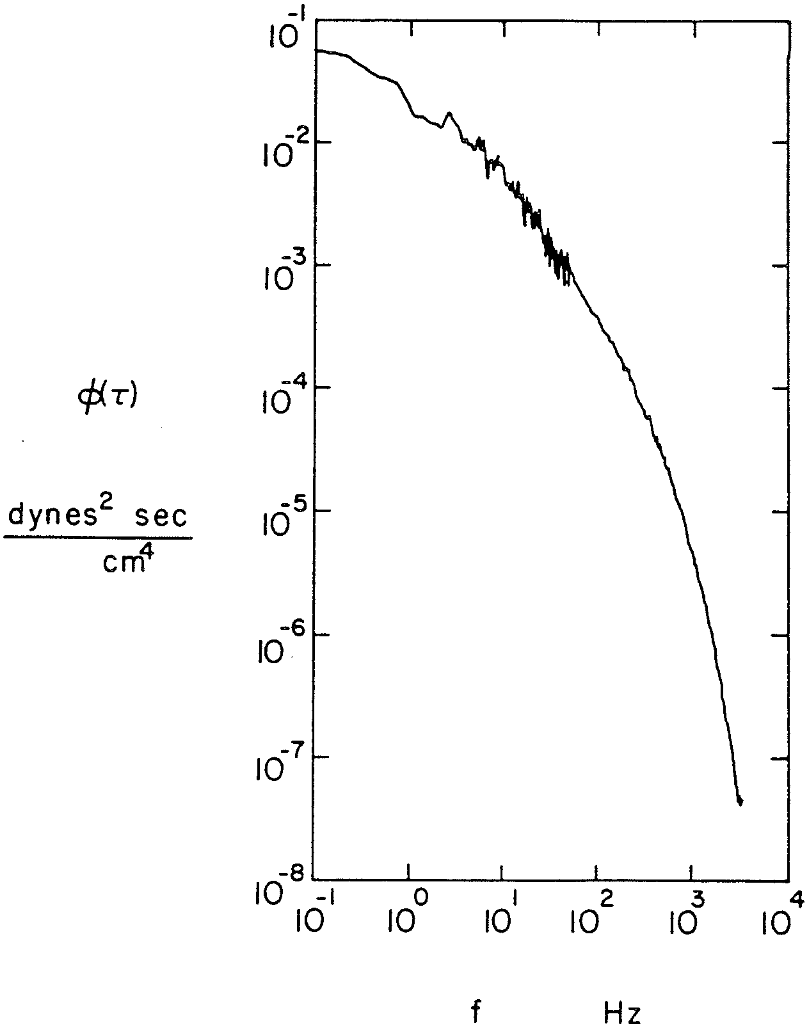


Figure 4.12: The τ spectrum at x=31.55 cm.

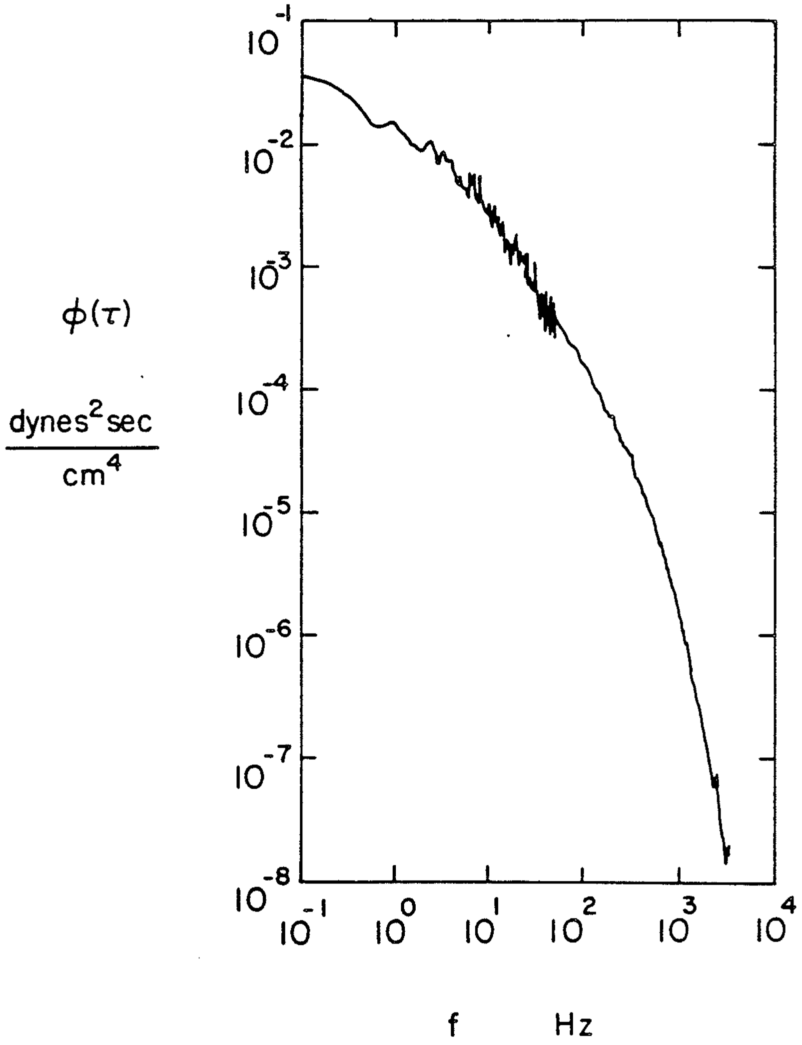


Figure 4.13: the τ spectrum at x=39.53 cm.

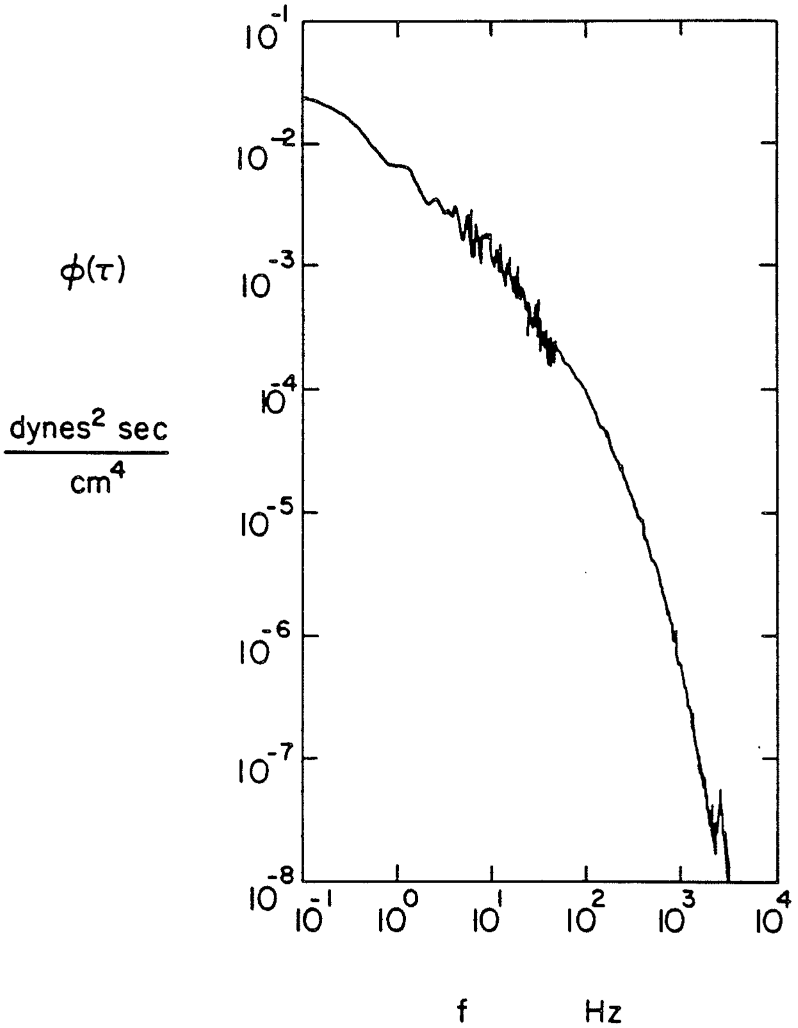


Figure 4.14: The τ spectrum at x=50.71 cm.

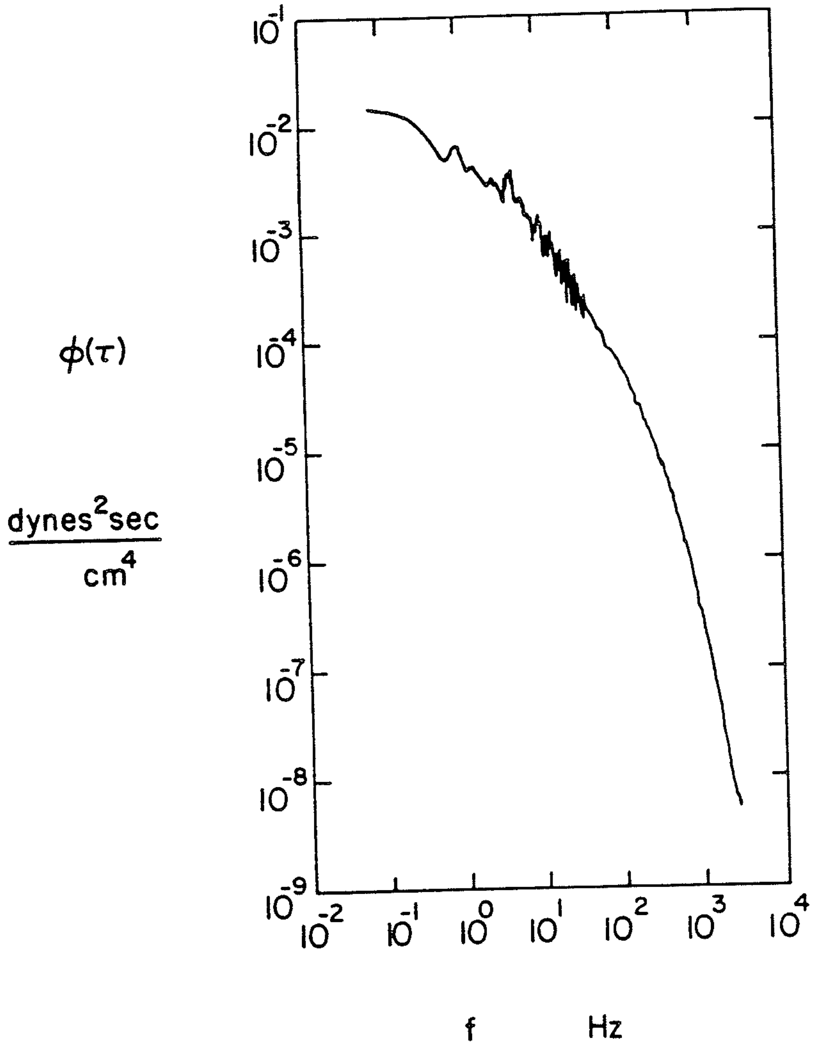


Figure 4.15: The τ spectrum at x=55.69 cm.

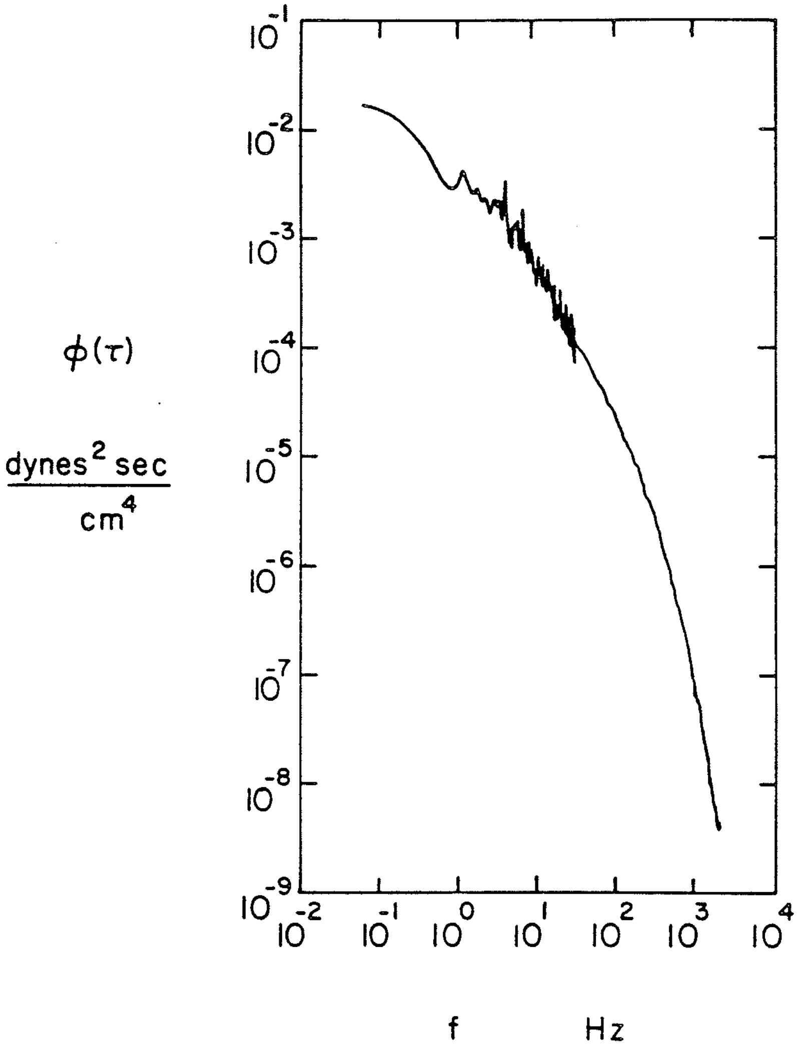


Figure 4.16: The τ spectrum at x=69.46 cm.

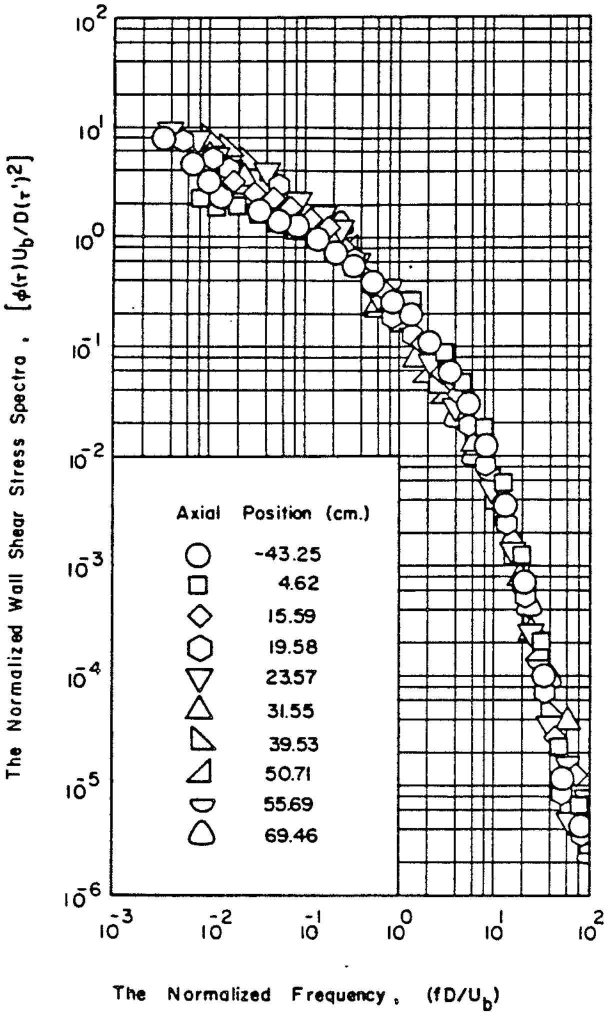


Figure 4.17: The nondimensionalized τ spectrum.

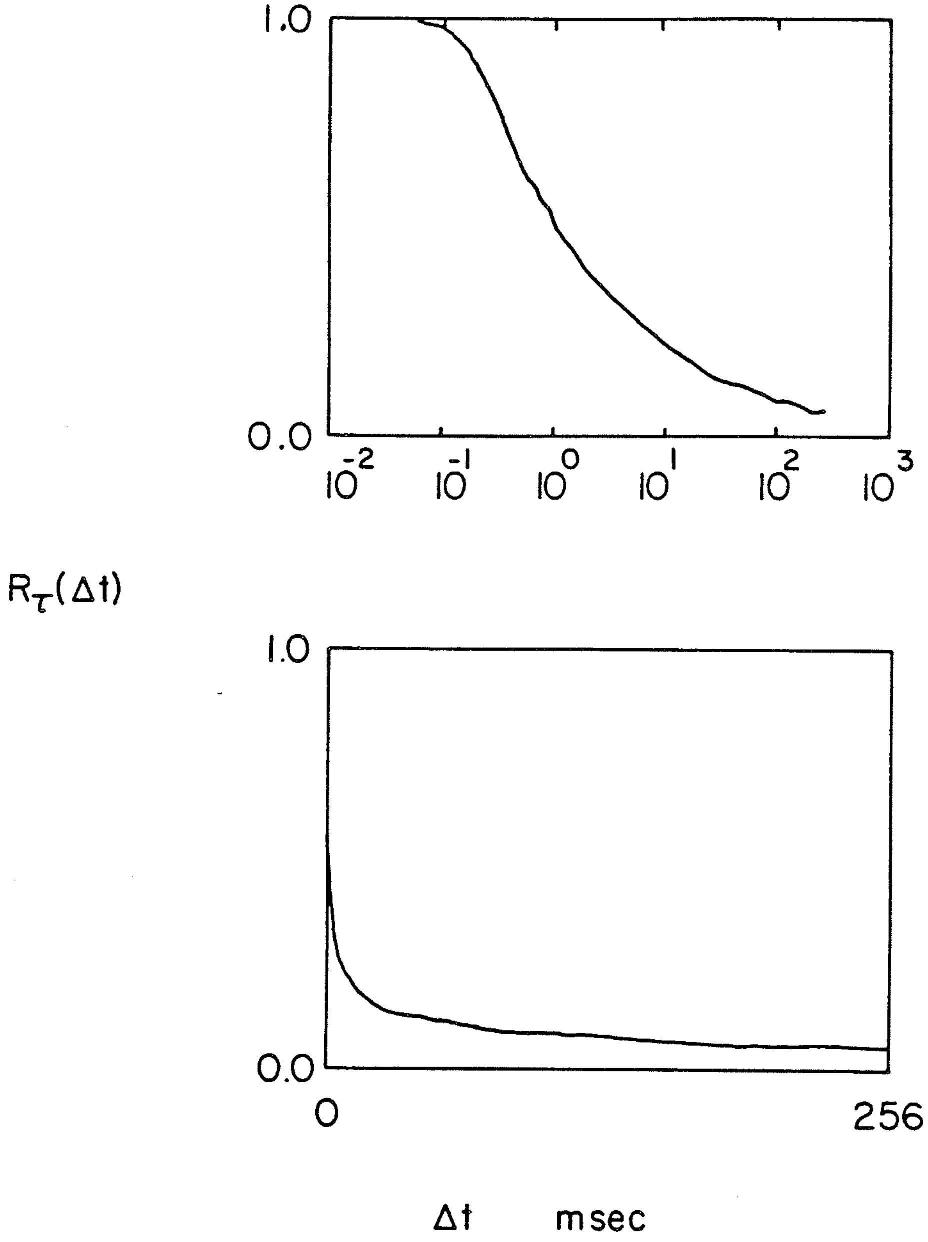


Figure 4.18: The autocorrelation of τ in the pipe.

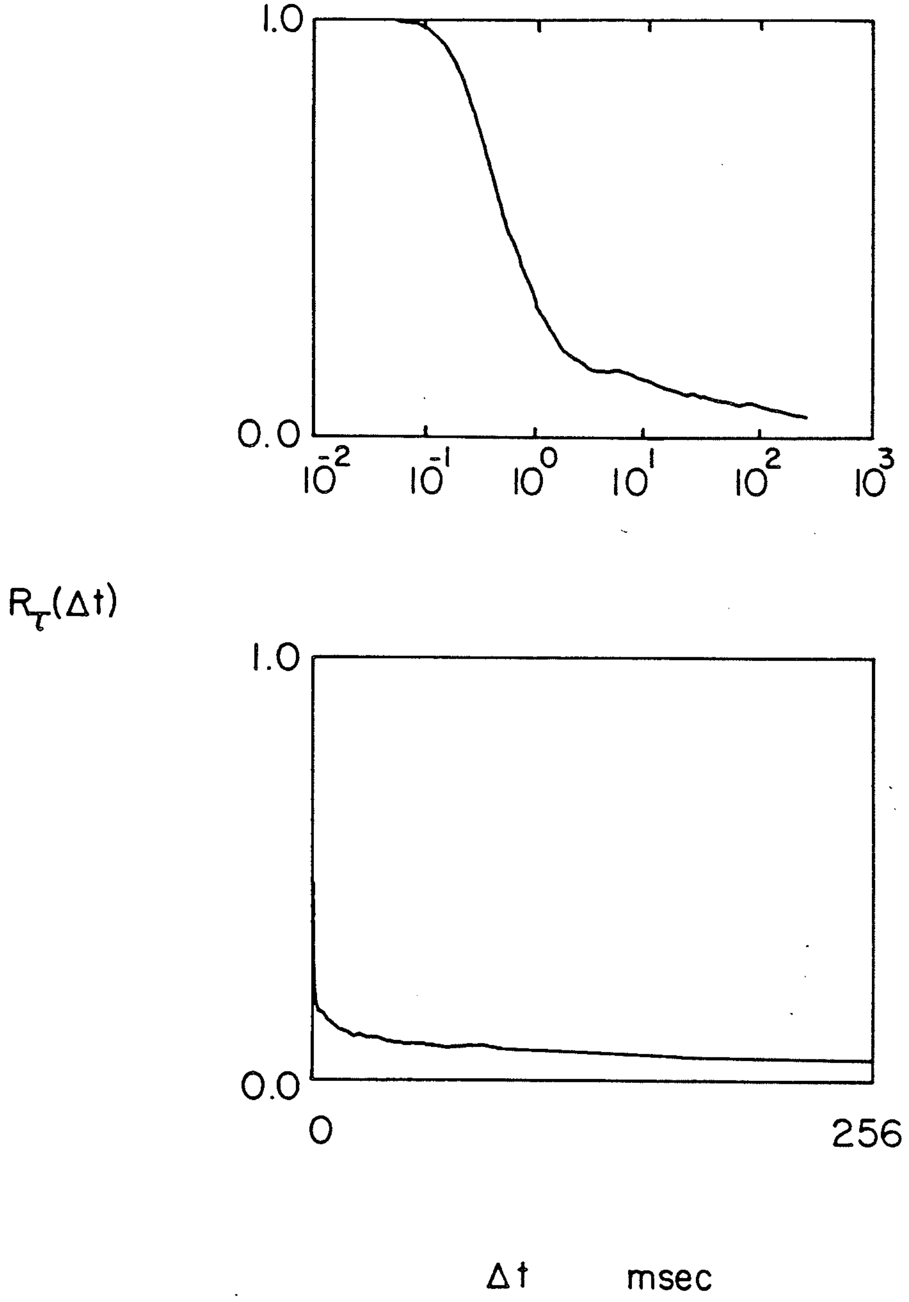


Figure 4.19: The autocorrelation of τ at x=4.62 cm.

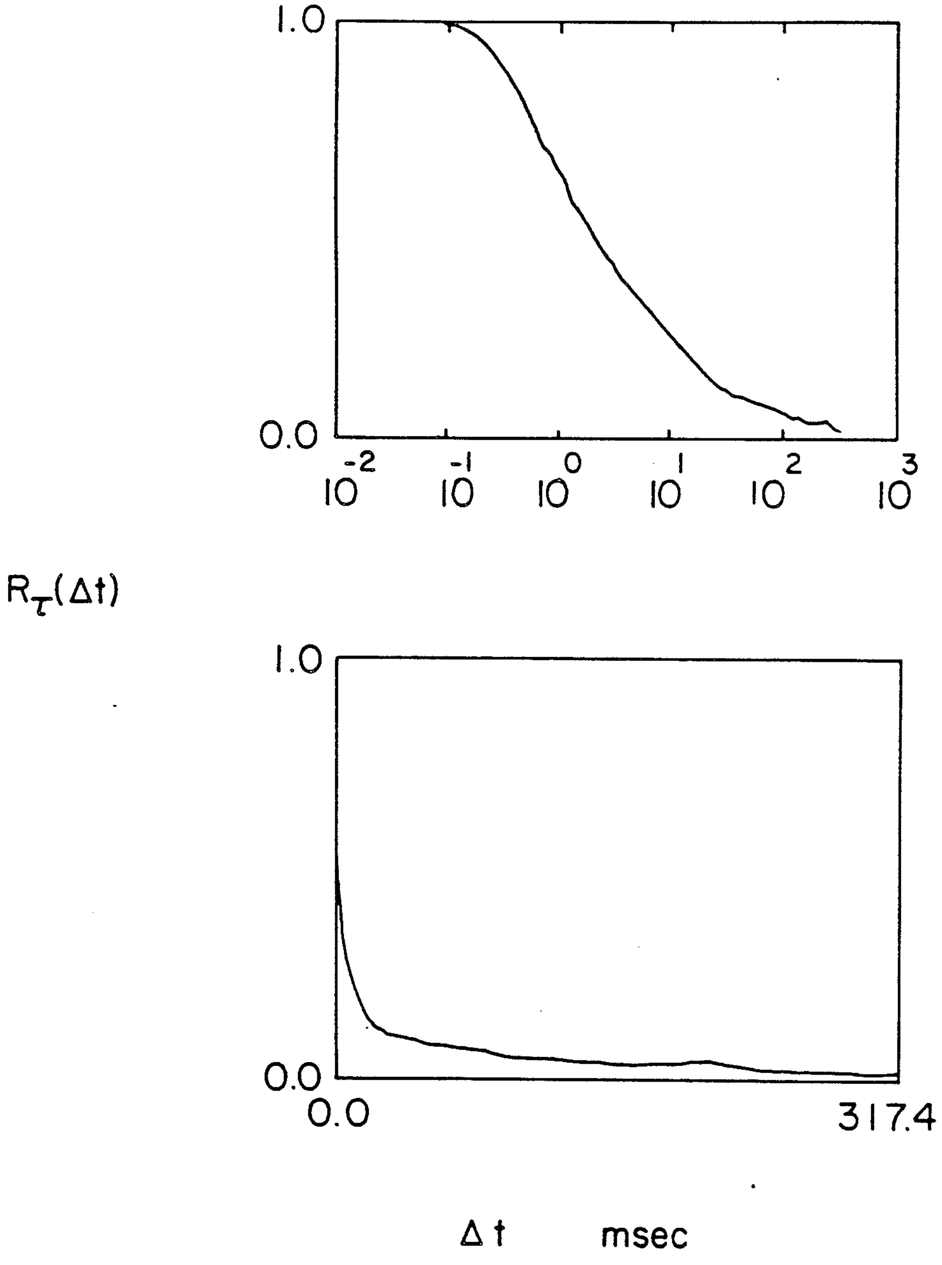


Figure 4.20: The autocorrelation of τ at x=15.59 cm.

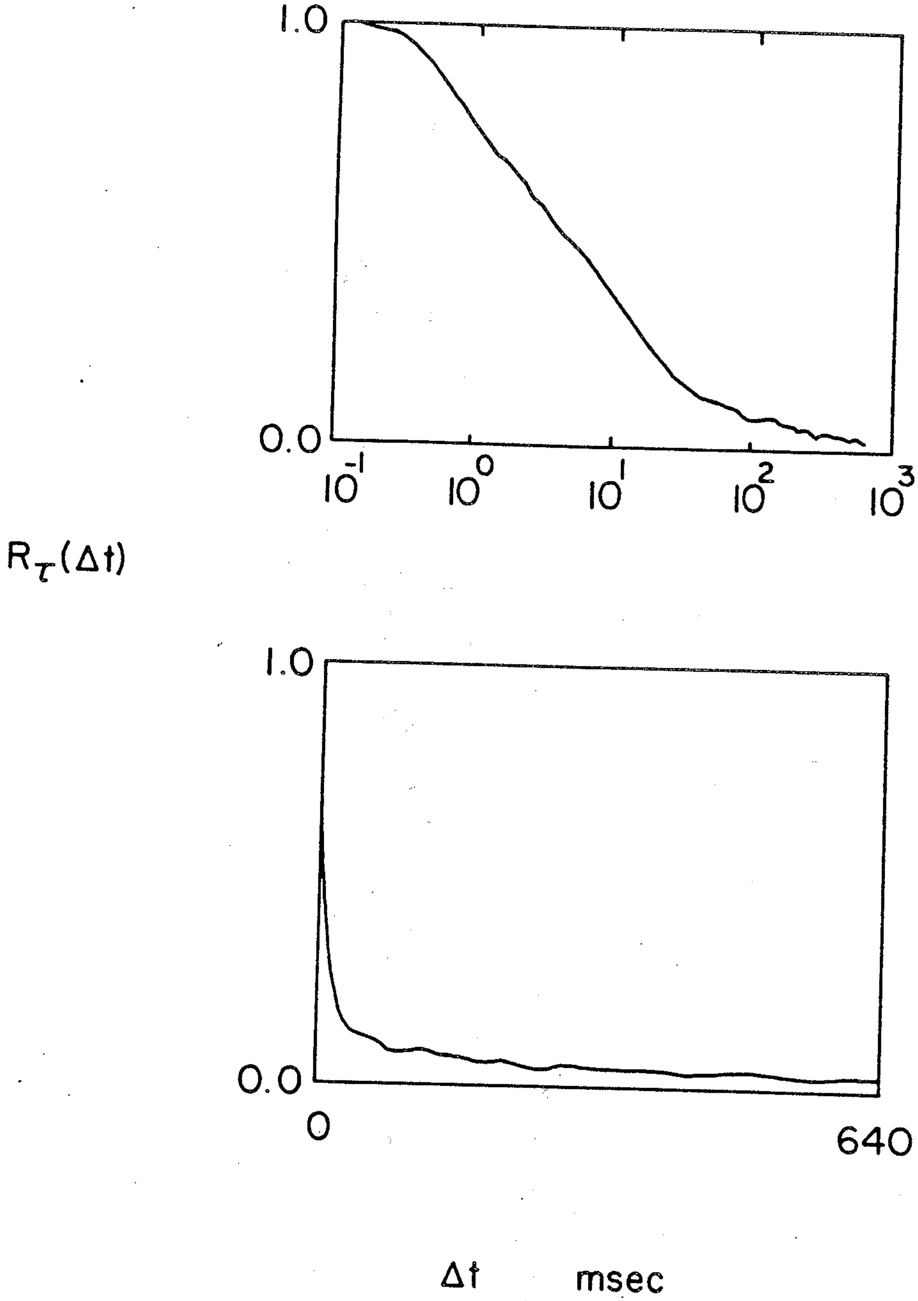


Figure 4.21: The autocorrelation of τ at x=19.58 cm.

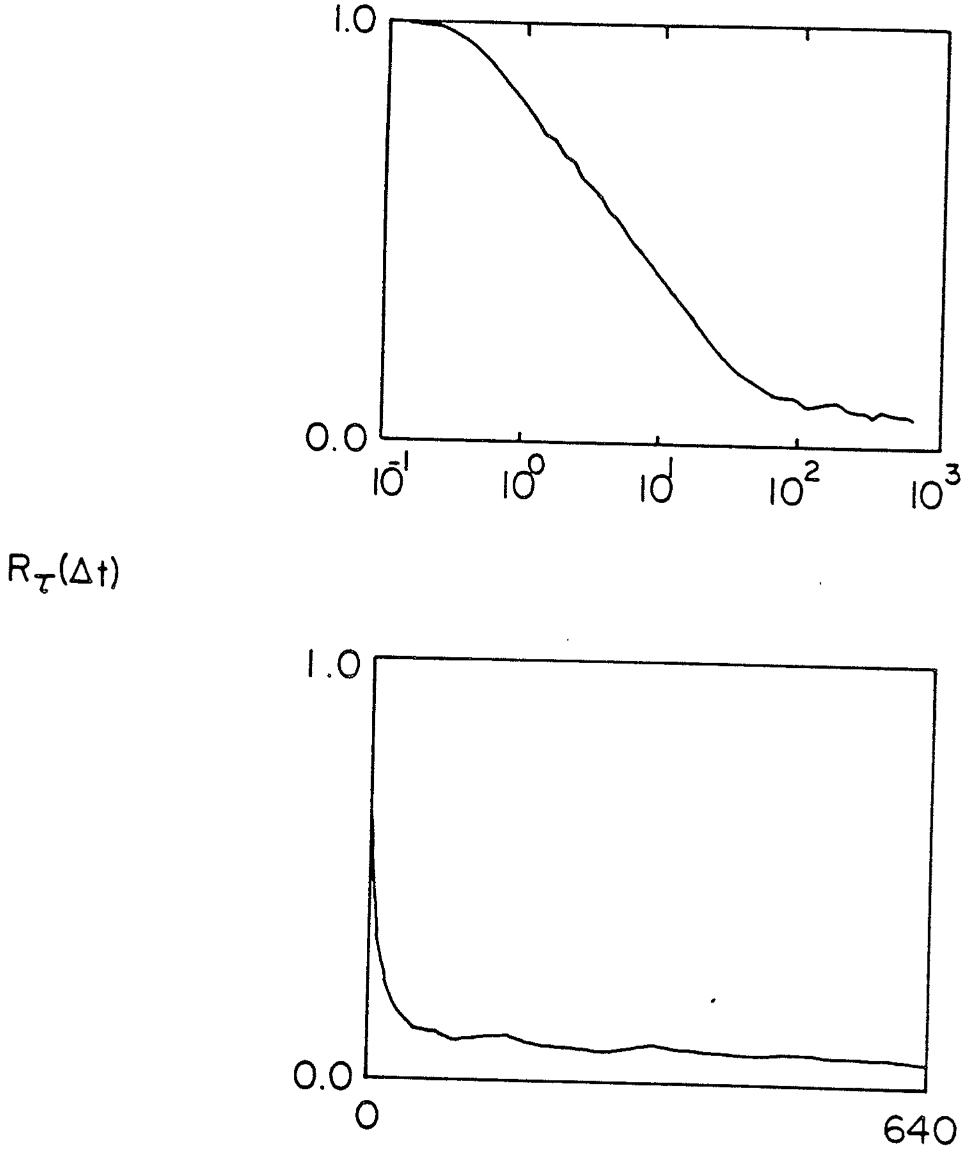


Figure 4.22: the autocorrelation of τ at x=23.57 cm.

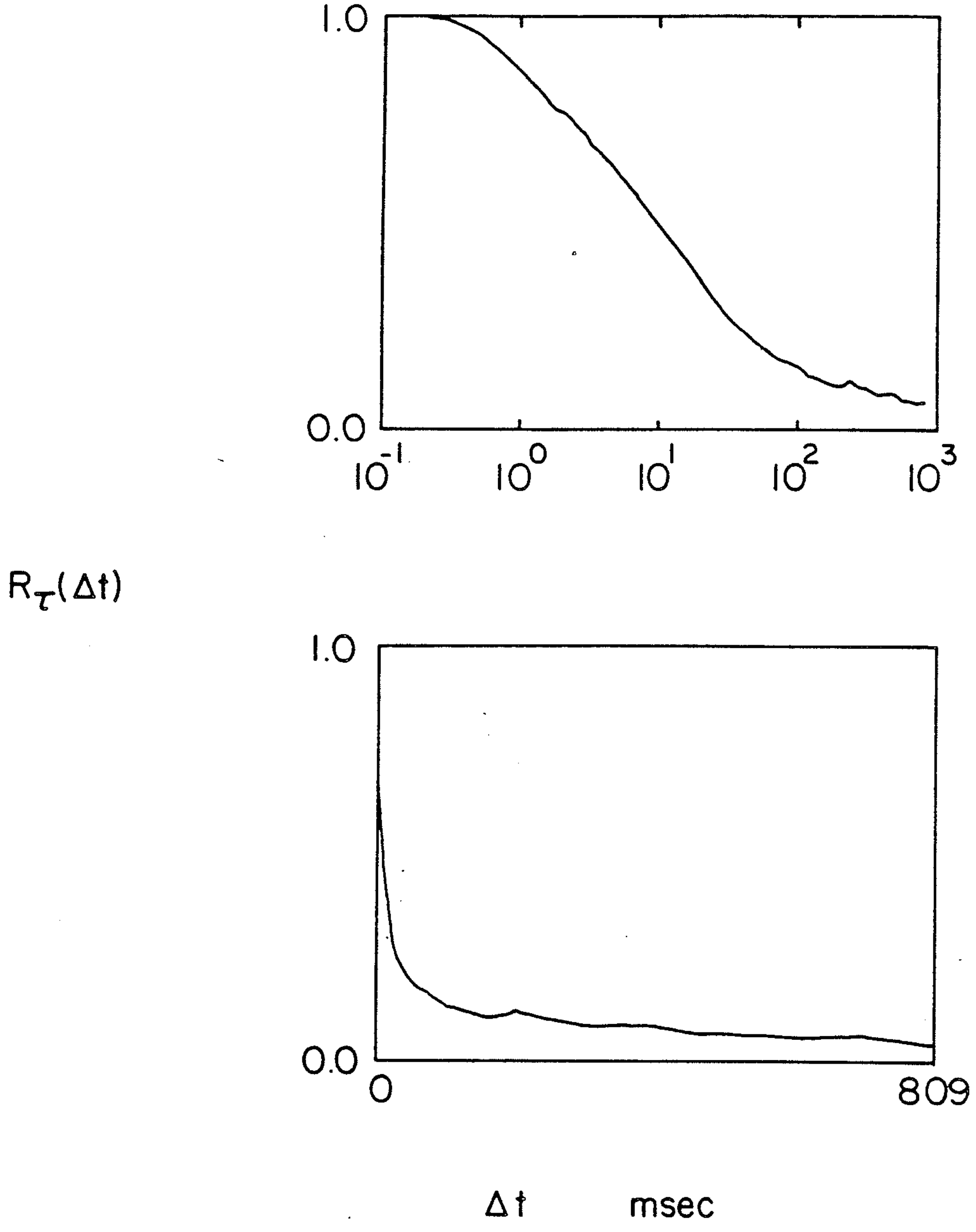


Figure 4.23: The autocorrelation of τ at x=31.55 cm.

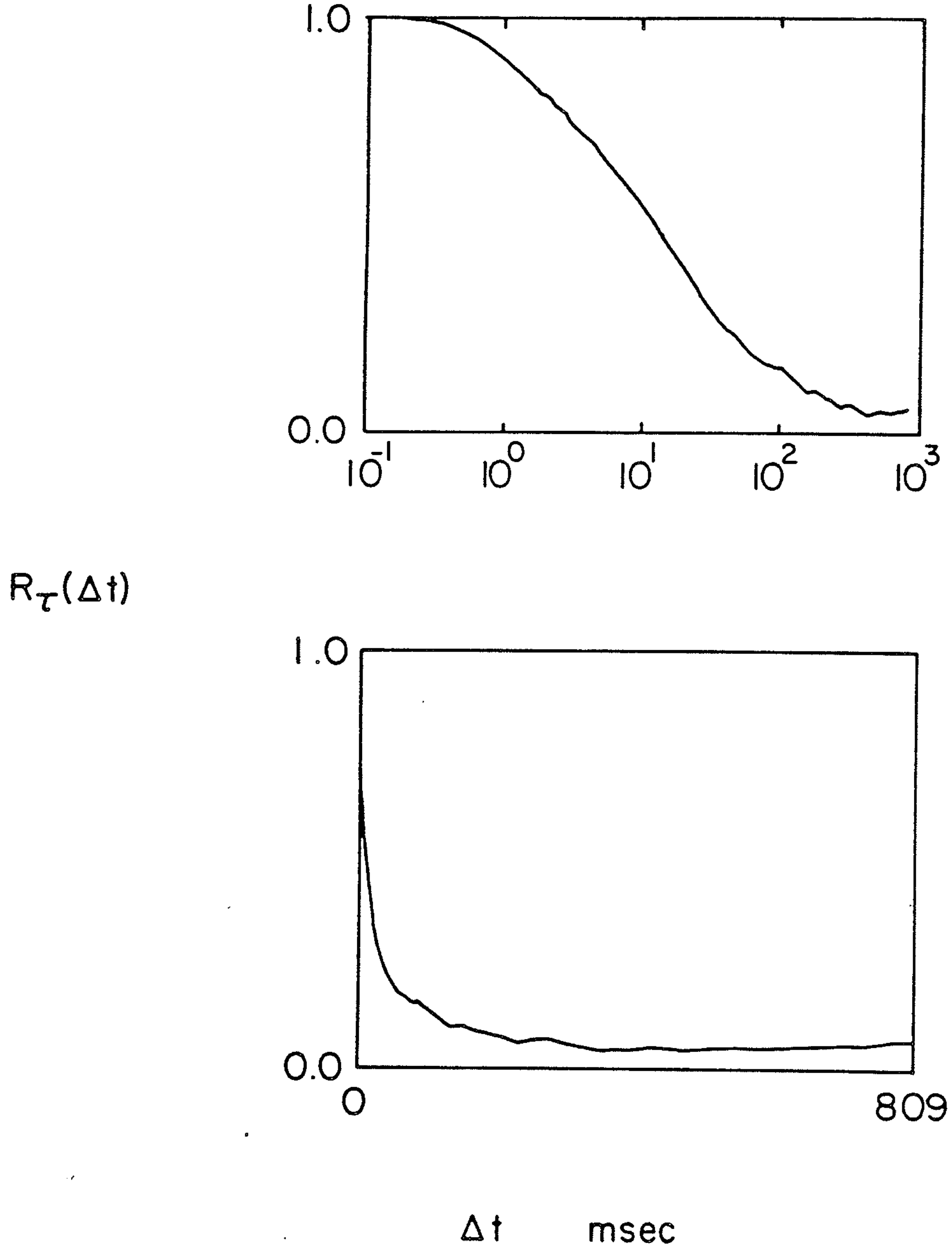


Figure 4.24: The autocorrelation of τ at x=39.53 cm.

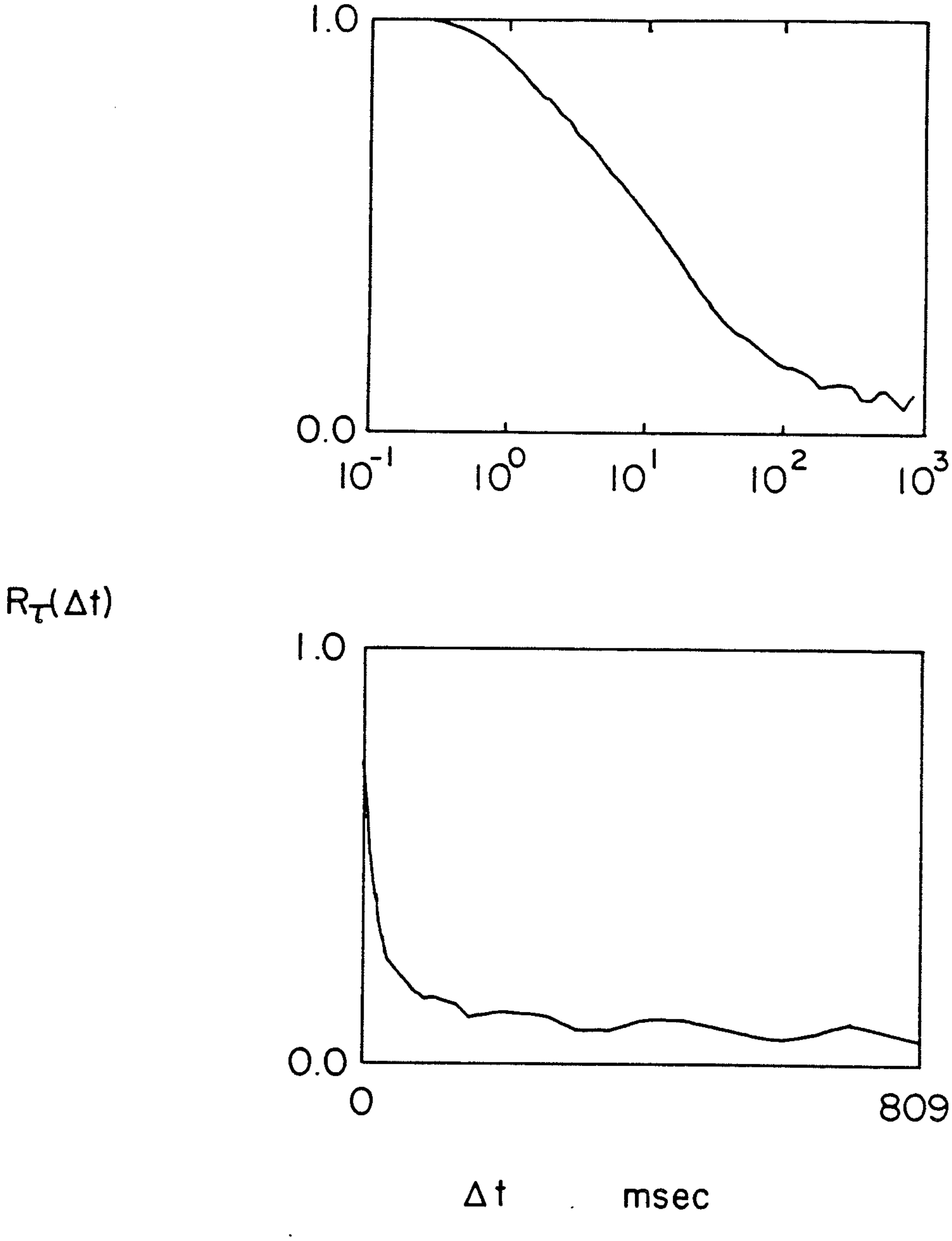


Figure 4.25: The autocorrelation of τ at x=50.71 cm.

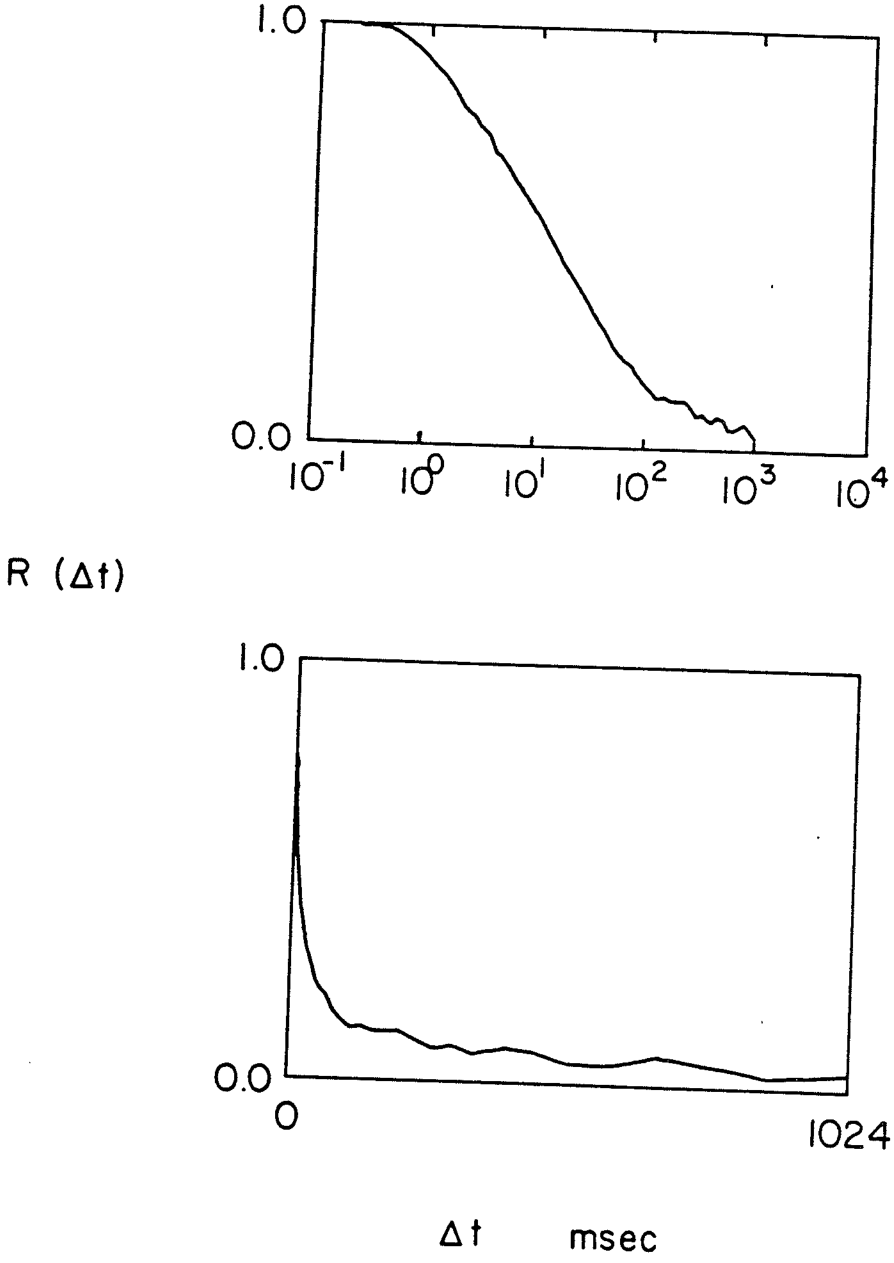


Figure 4.26: The autocorrelation of τ at x=55.69 cm.

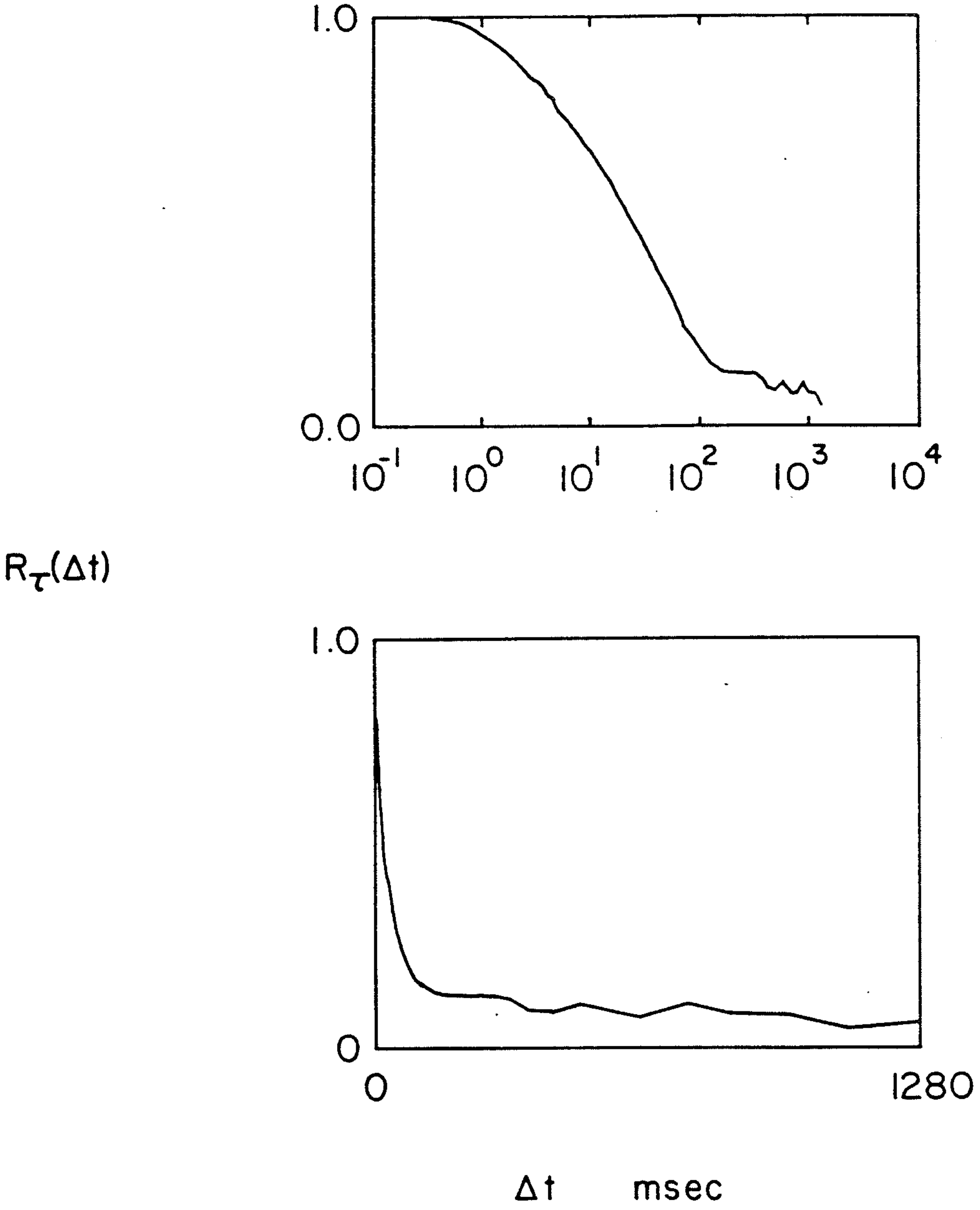


Figure 4.27: The autocorrelation of τ at x=69.46 cm.

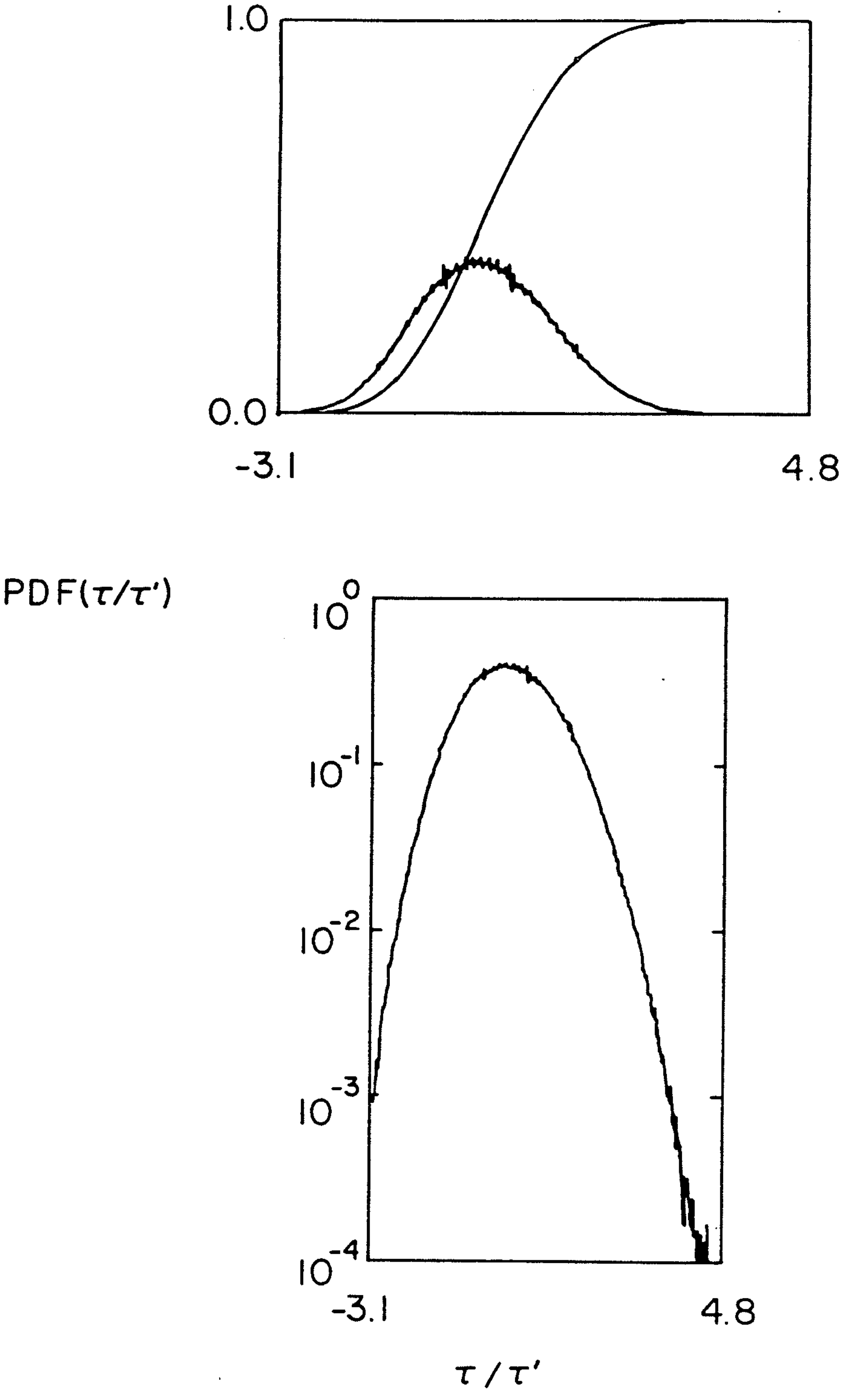


Figure 4.28: The PDF(τ/τ') in the pipe.

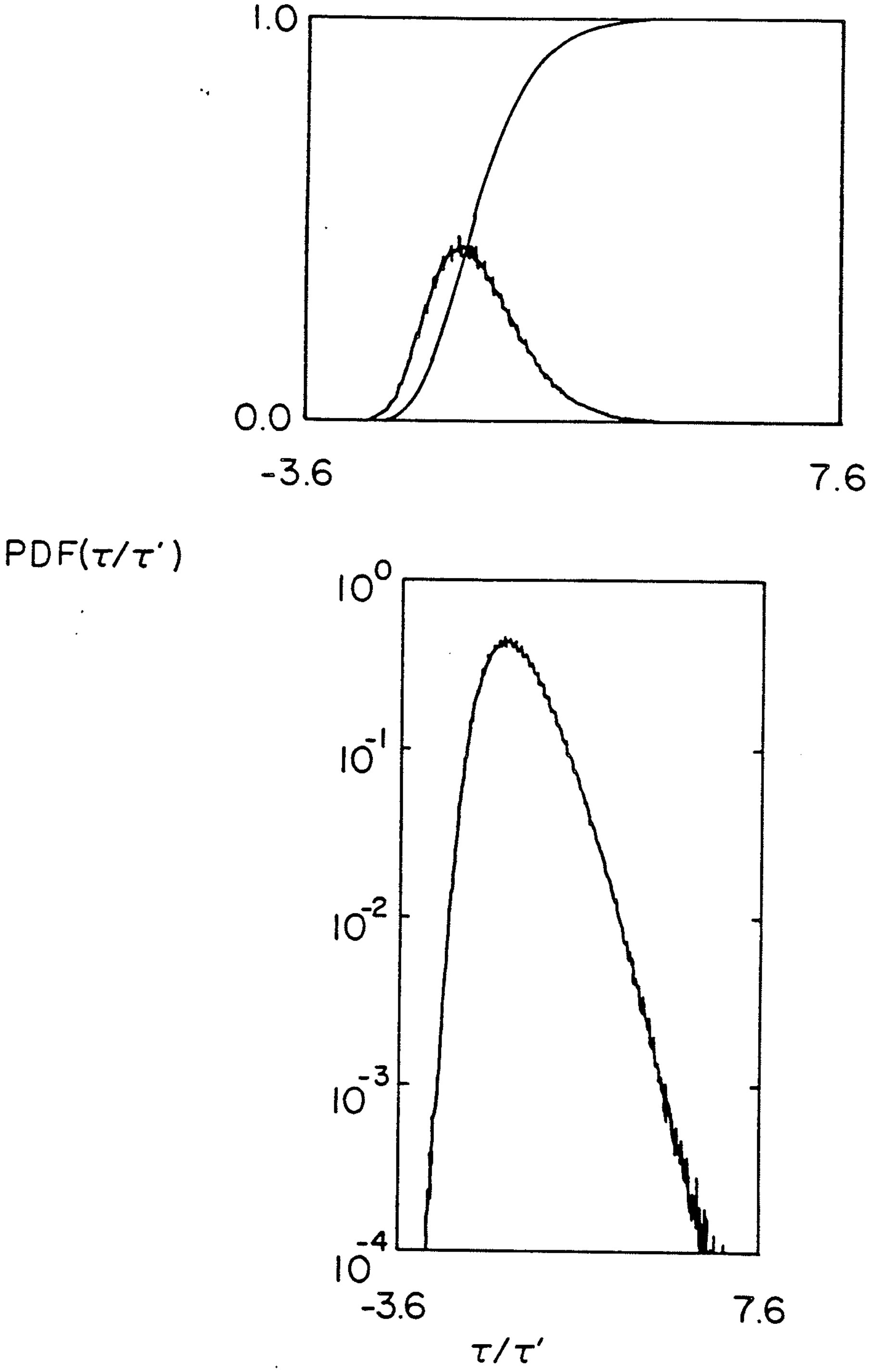


Figure 4.29: The PDF(τ/τ') at x=4.62 cm.

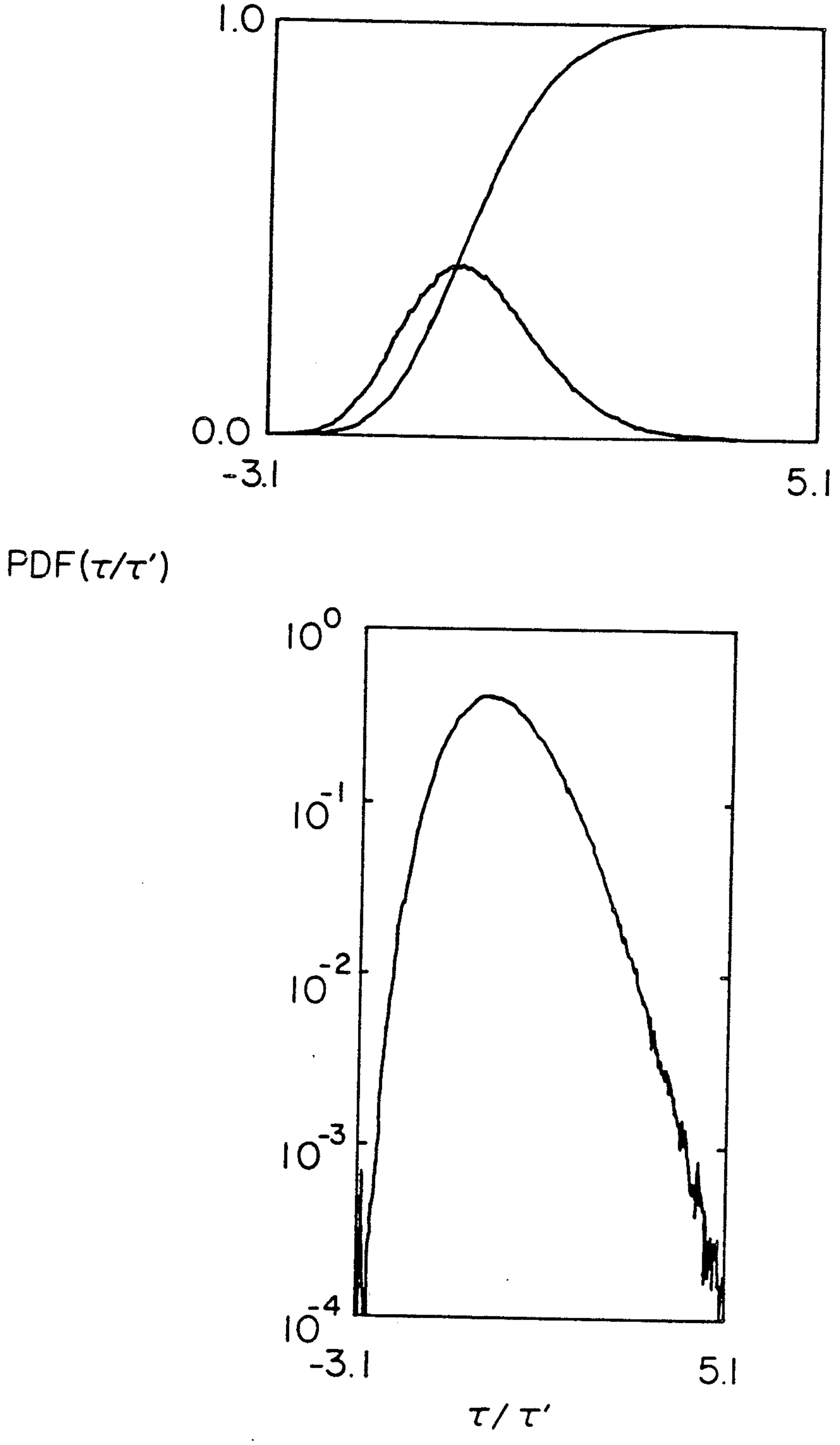


Figure 4.30: The PDF(τ/τ') at x=15.59 cm.

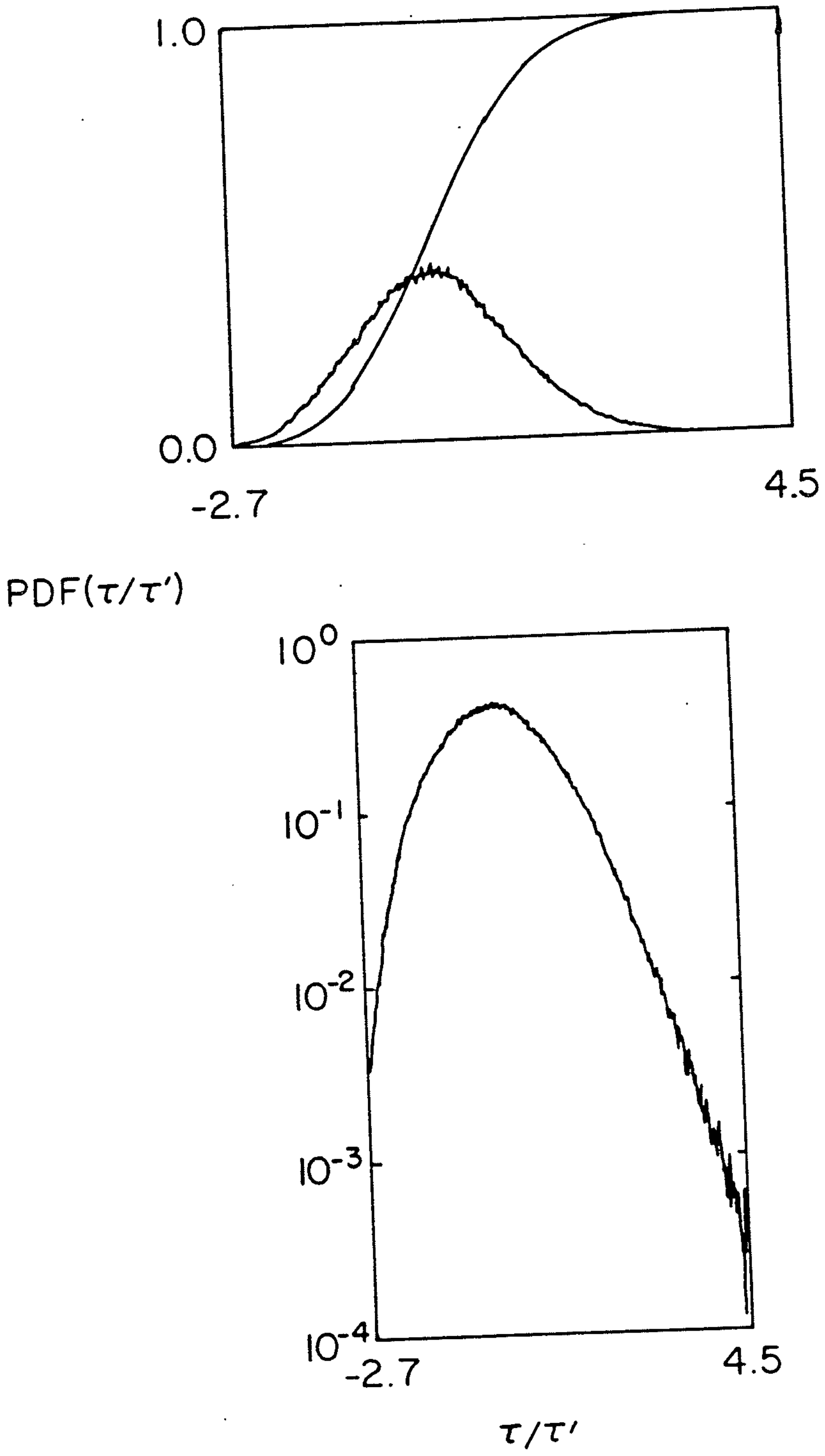


Figure 4.31: The PDF(τ/τ') at x=19.58 cm.

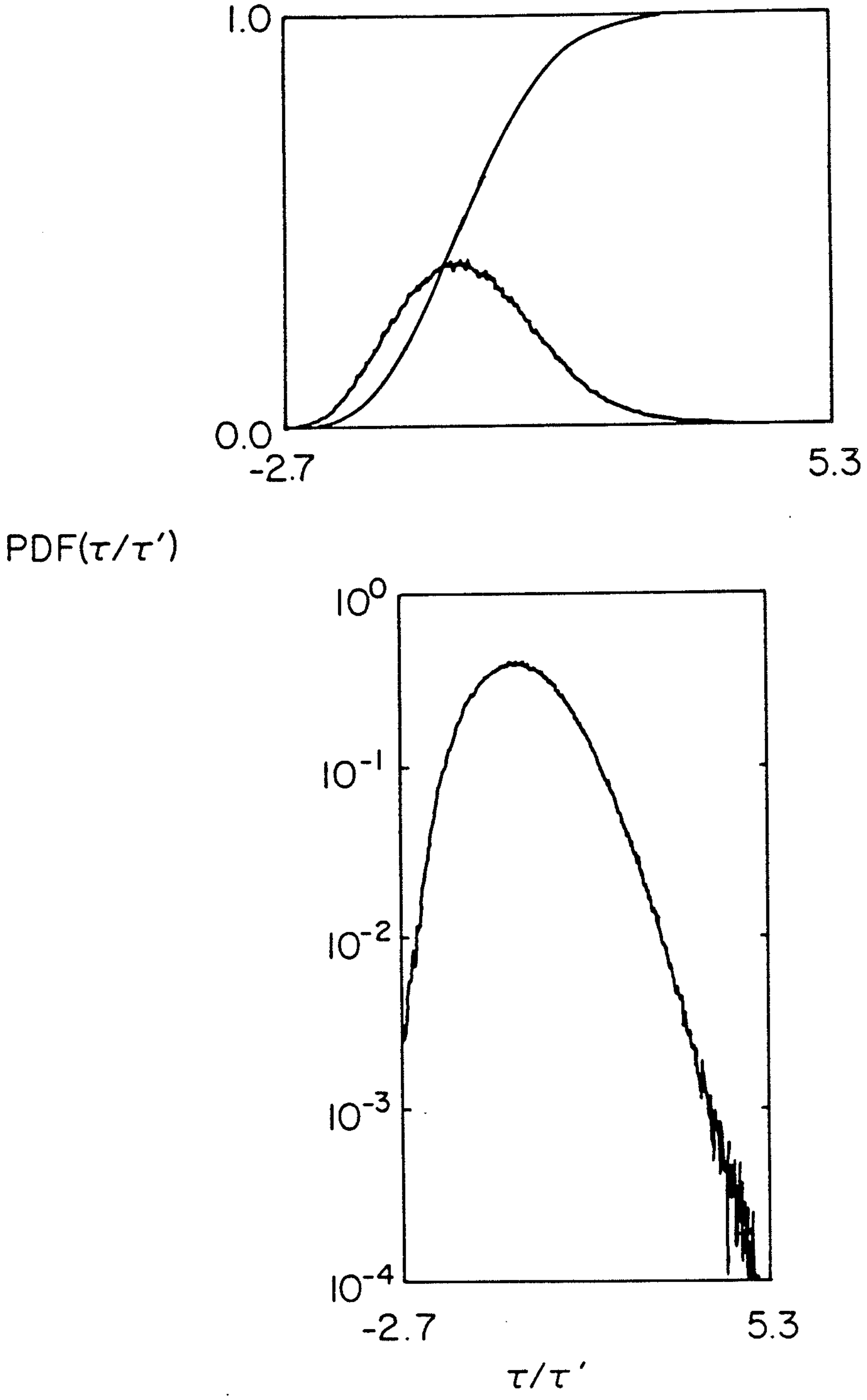


Figure 4.32: The PDF(τ/τ') at x=23.57 cm.

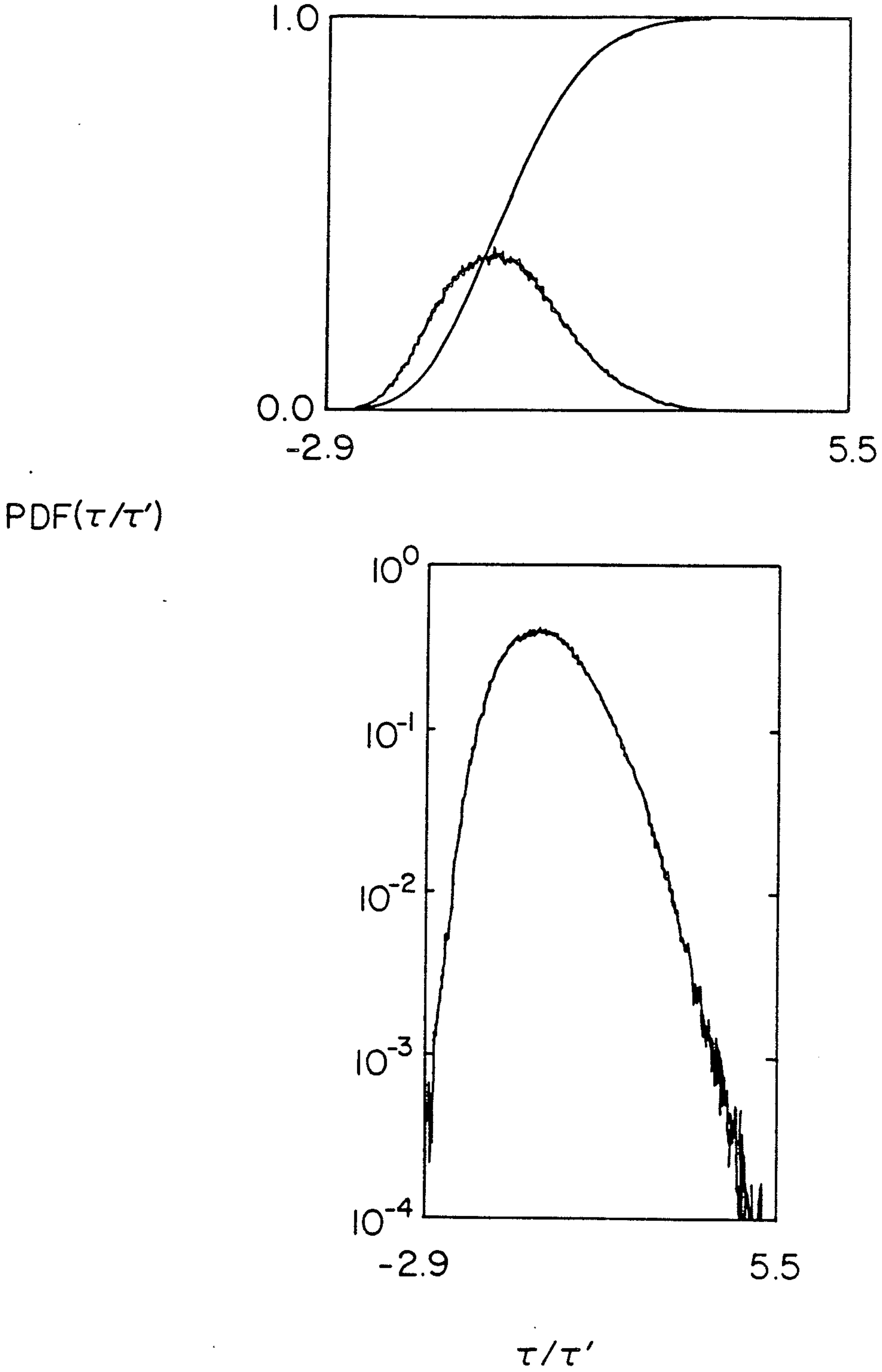


Figure 4.33: The PDF(τ/τ') at x=31.55 cm.

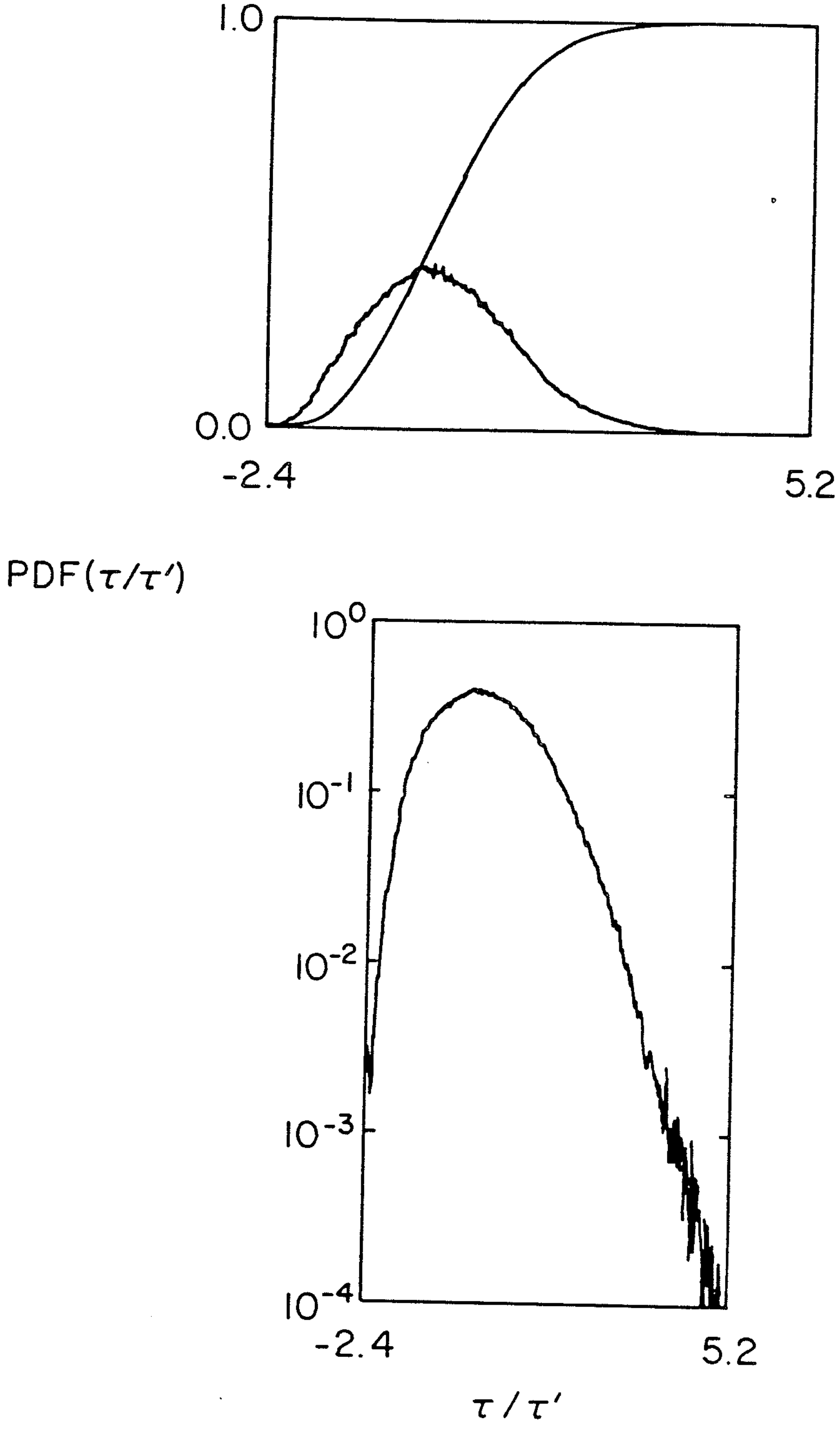


Figure 4.34: The PDF(τ/τ') at x=39.53 cm.

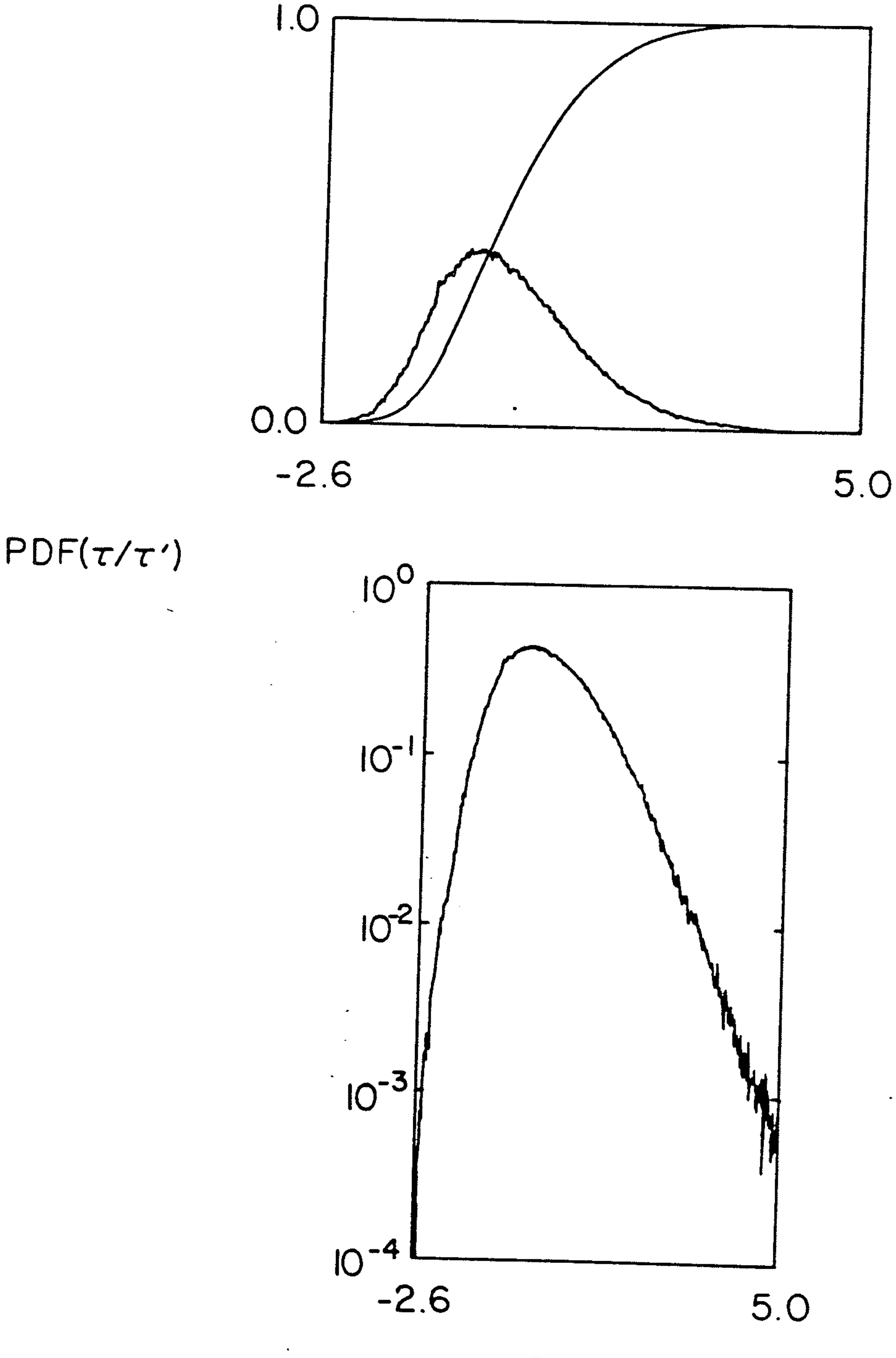


Figure 4.35: The PDF(τ/τ') at x=50.71 cm.

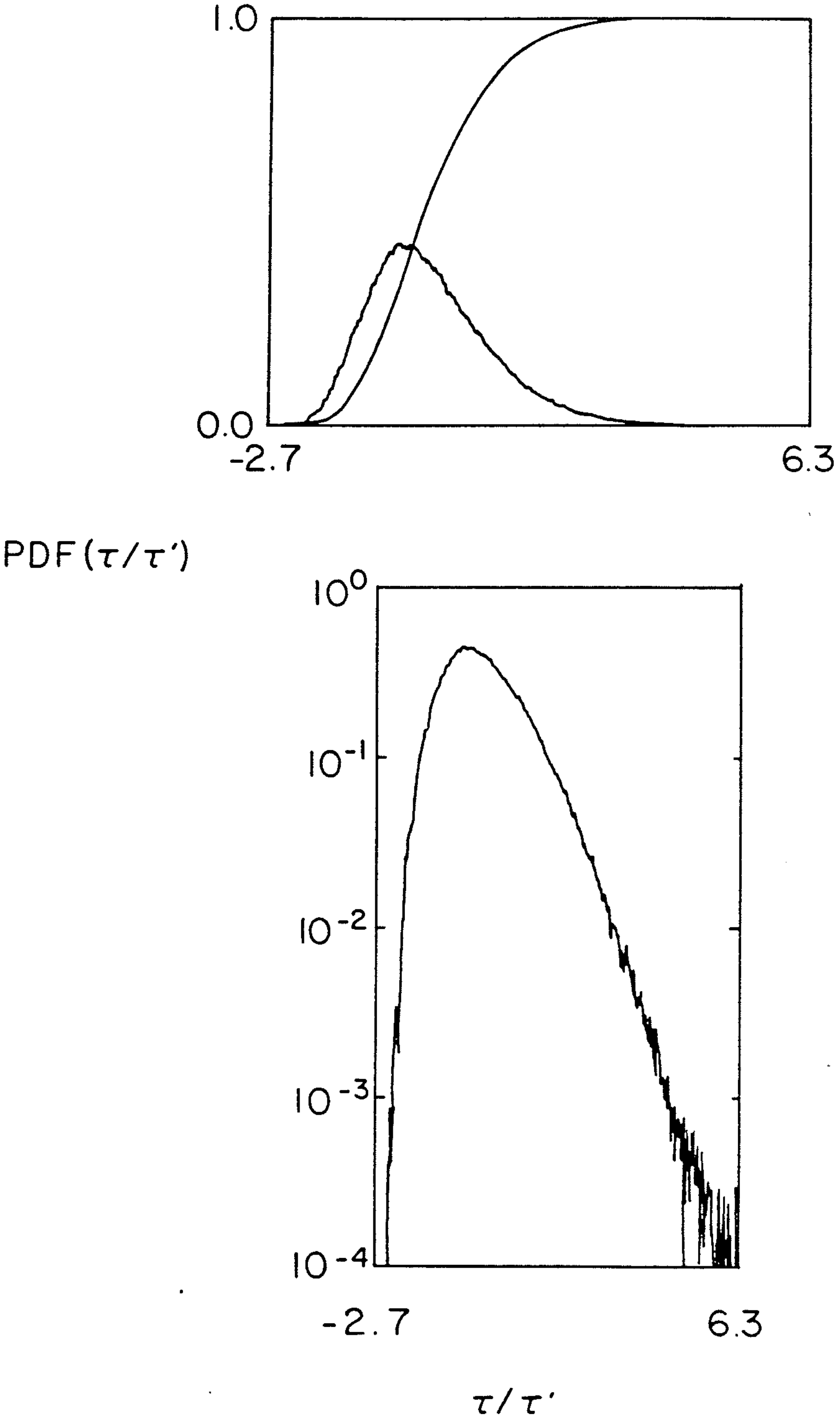


Figure 4.36: The PDF(τ/τ') at x=55.69 cm.

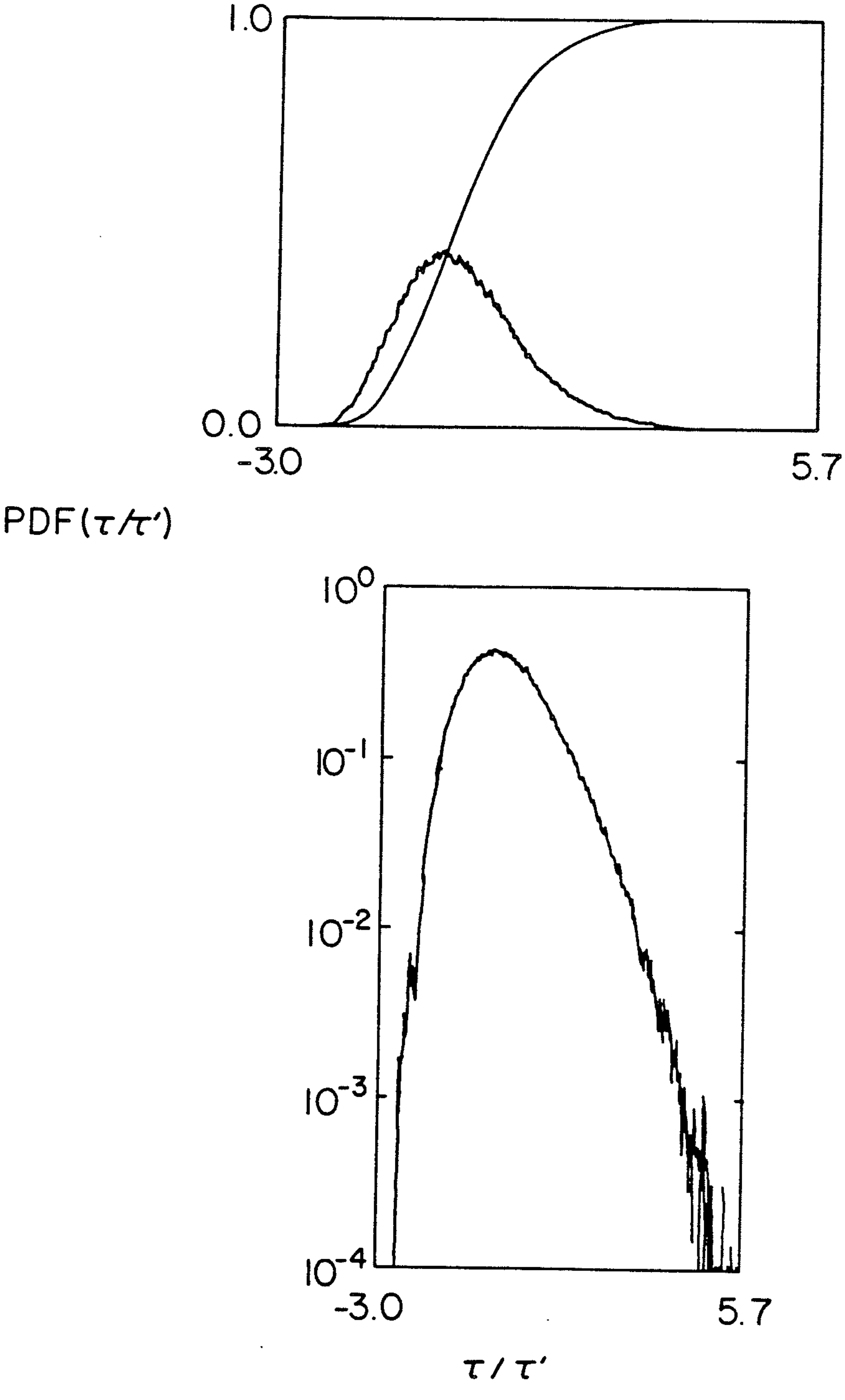


Figure 4.37: The PDF(τ/τ') at x=69.46 cm.

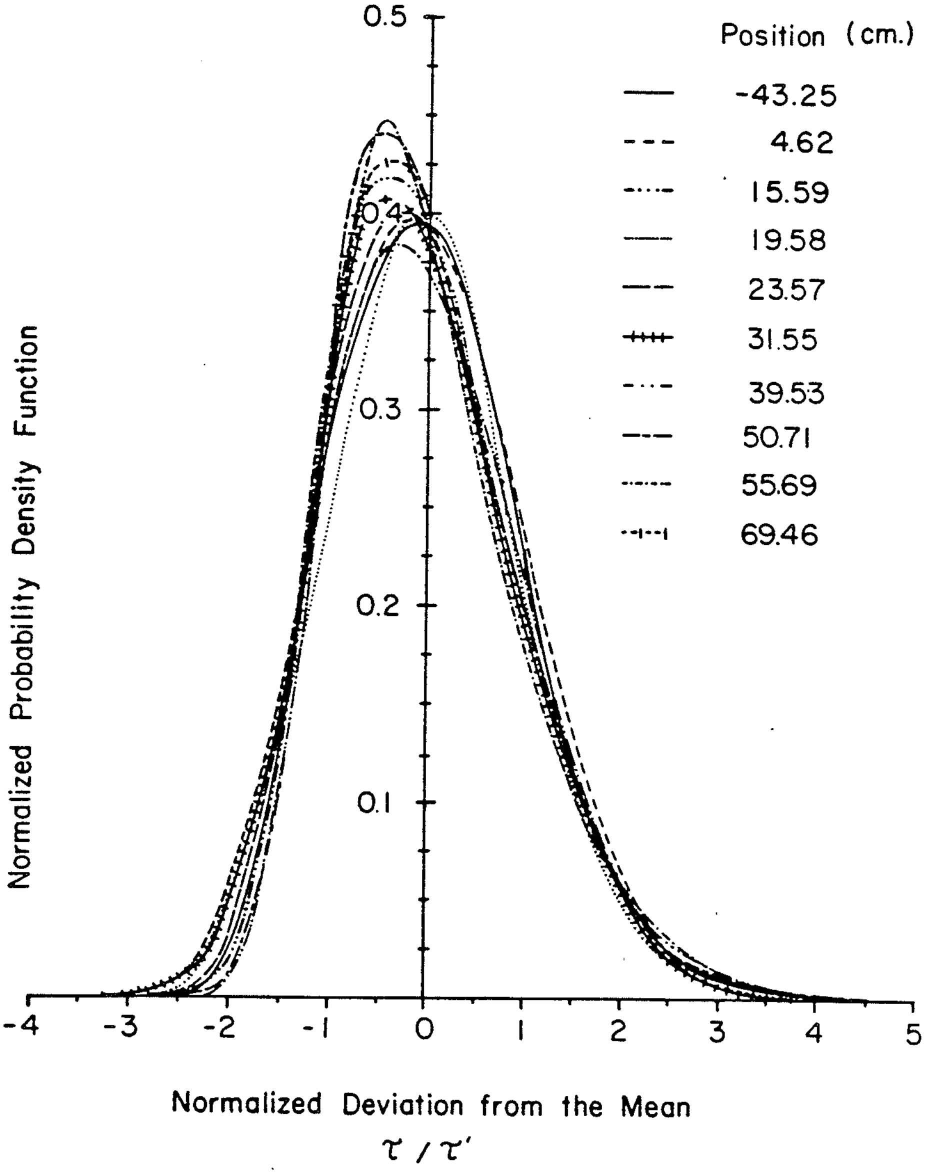


Figure 4.38: A compilation of the probability density functions.

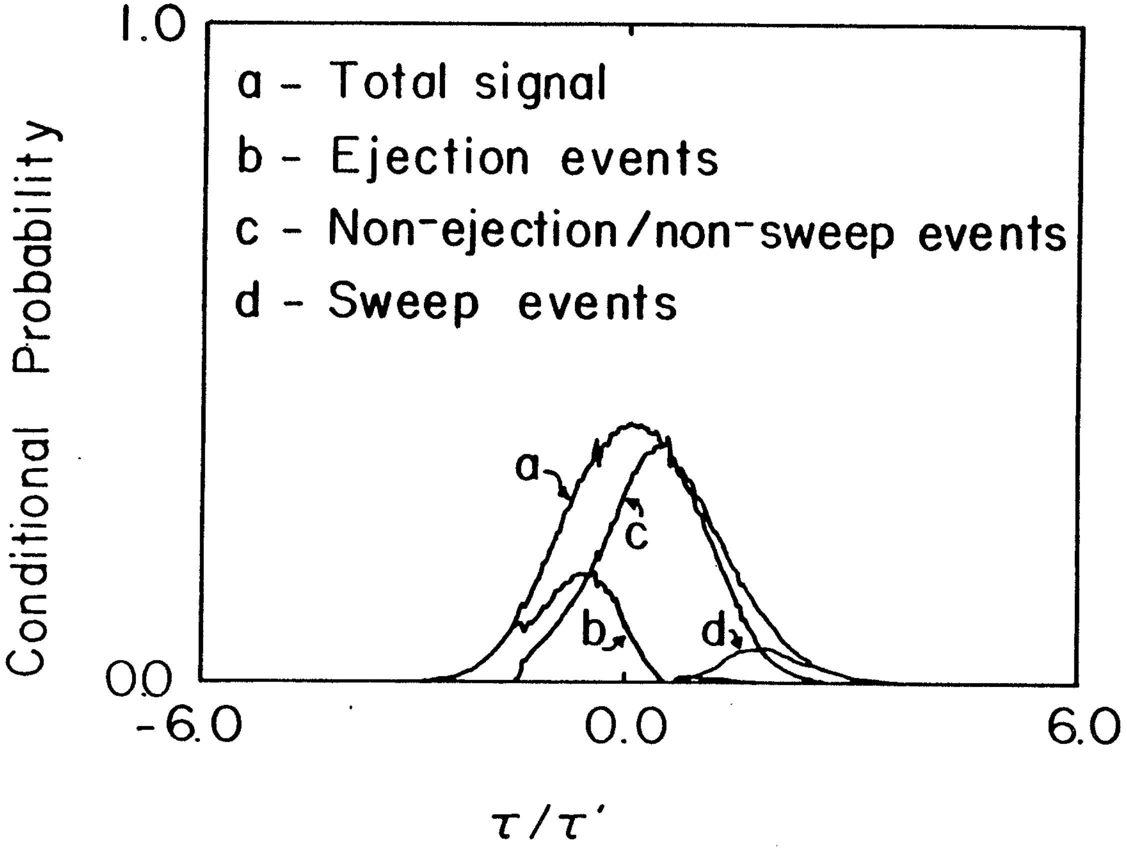


Figure 4.39: The Zaric decomposition of τ in the pipe.

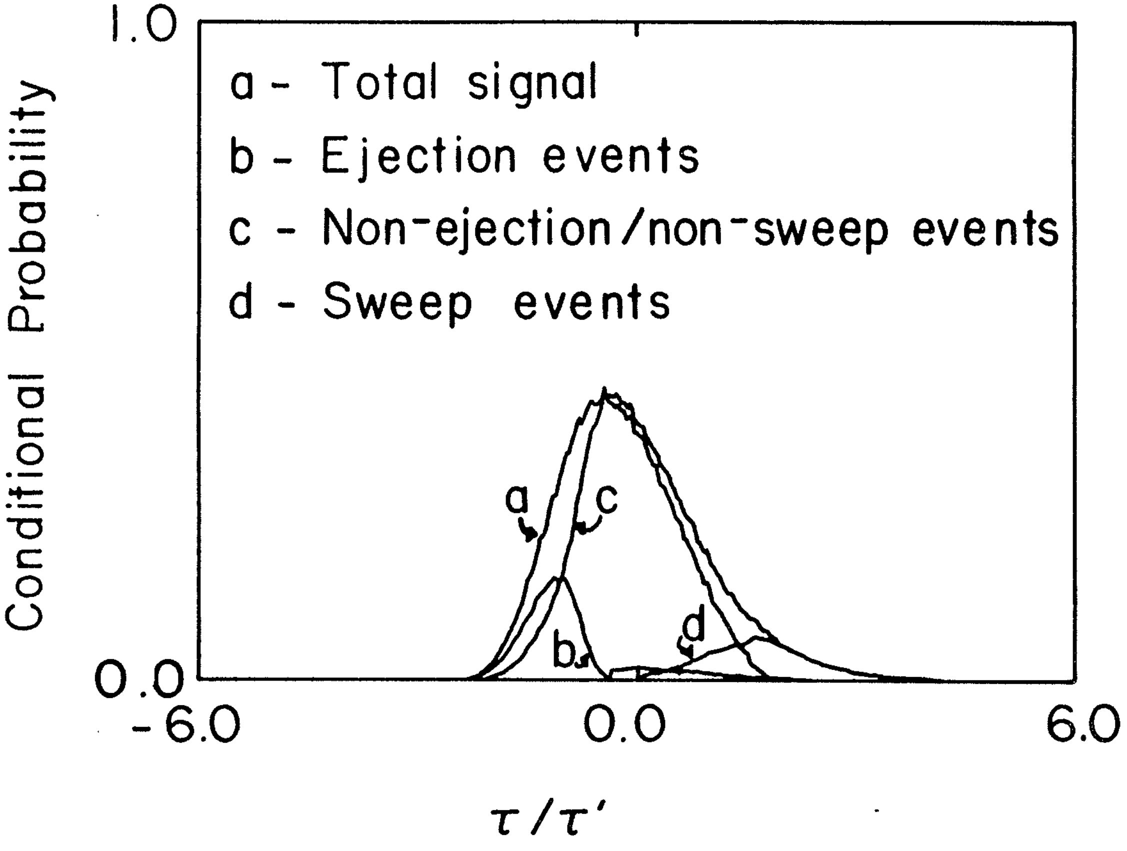


Figure 4.40: The Zaric decomposition of τ at x=4.62 cm.

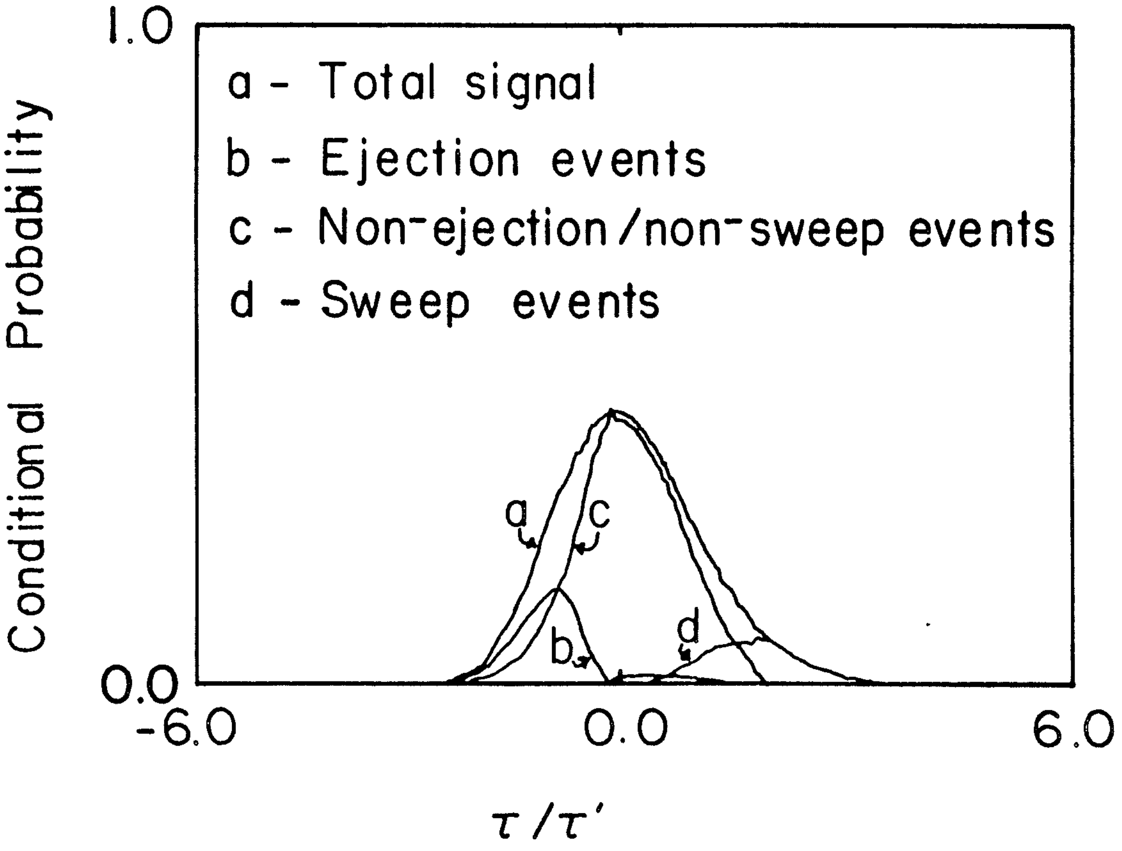


Figure 4.41: The Zaric decomposition of τ at x=15.59 cm.

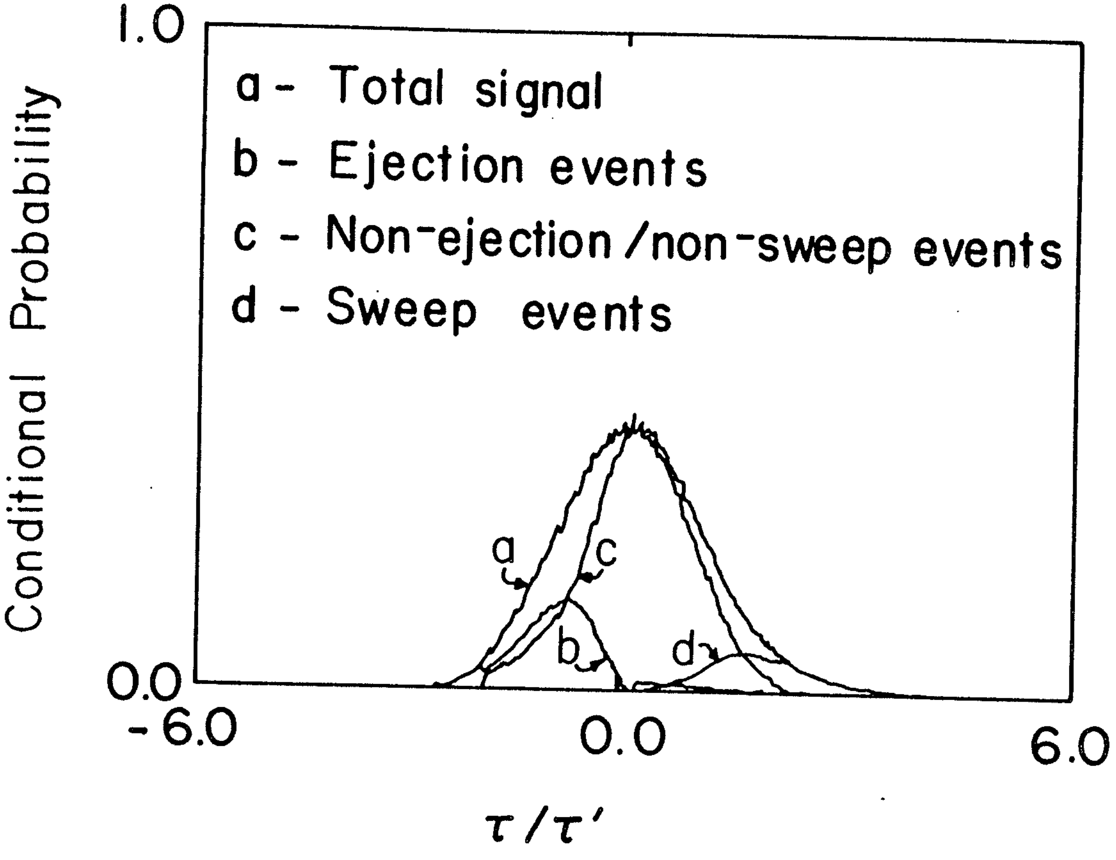


Figure 4.42: The Zaric decomposition of τ at x=19.58 cm.

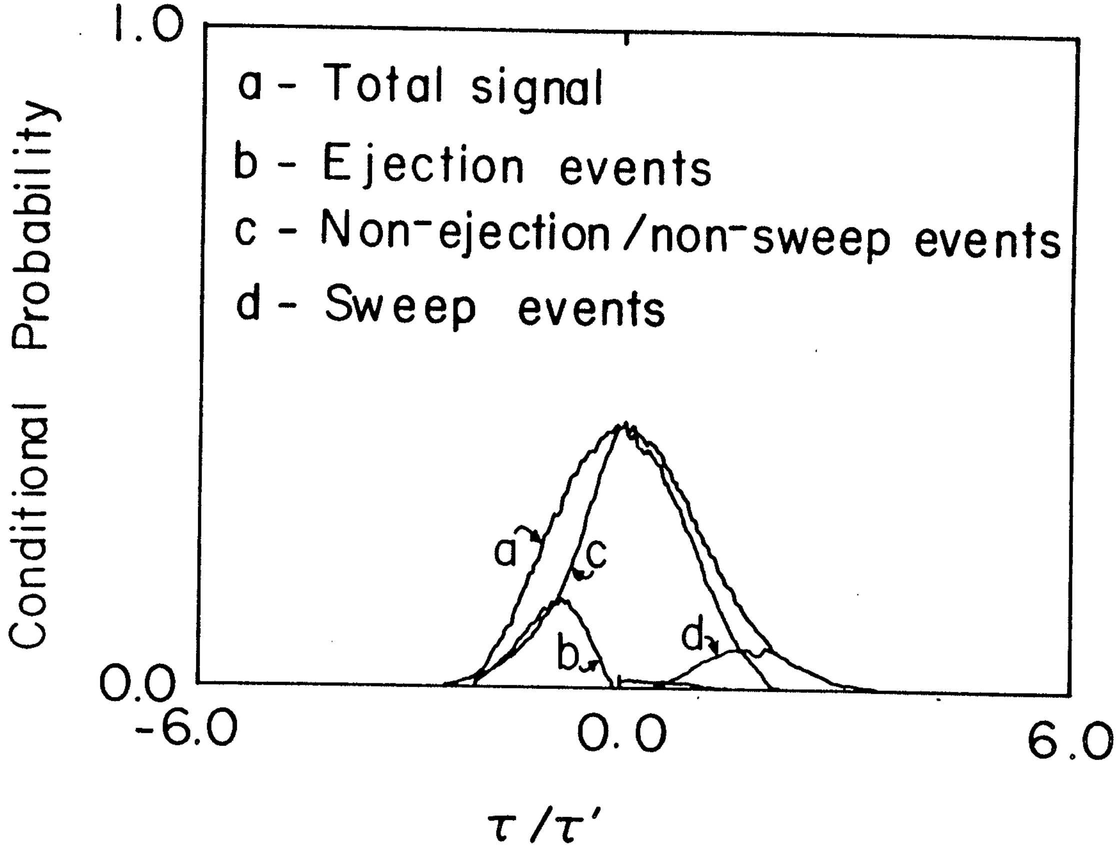


Figure 4.43: The Zaric decomposition of τ at x=23.57 cm.

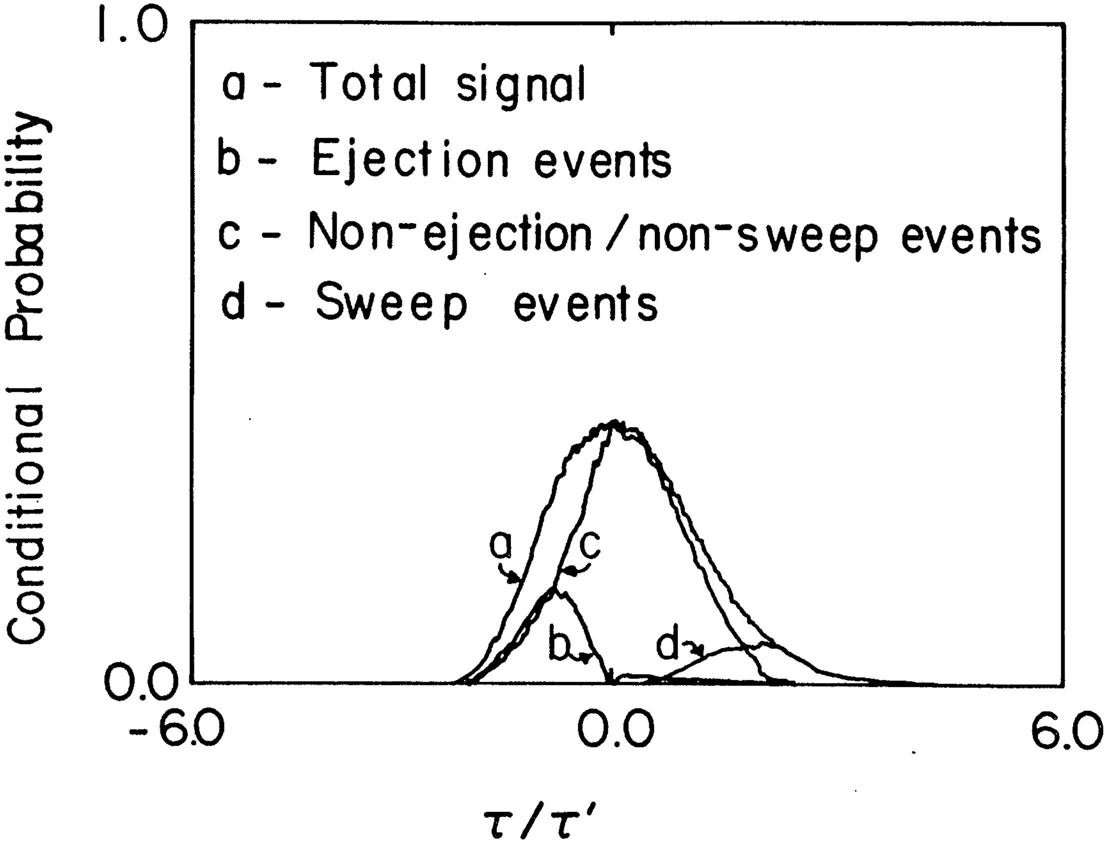


Figure 4.44: The Zaric decomposition of τ at x=31.55 cm.

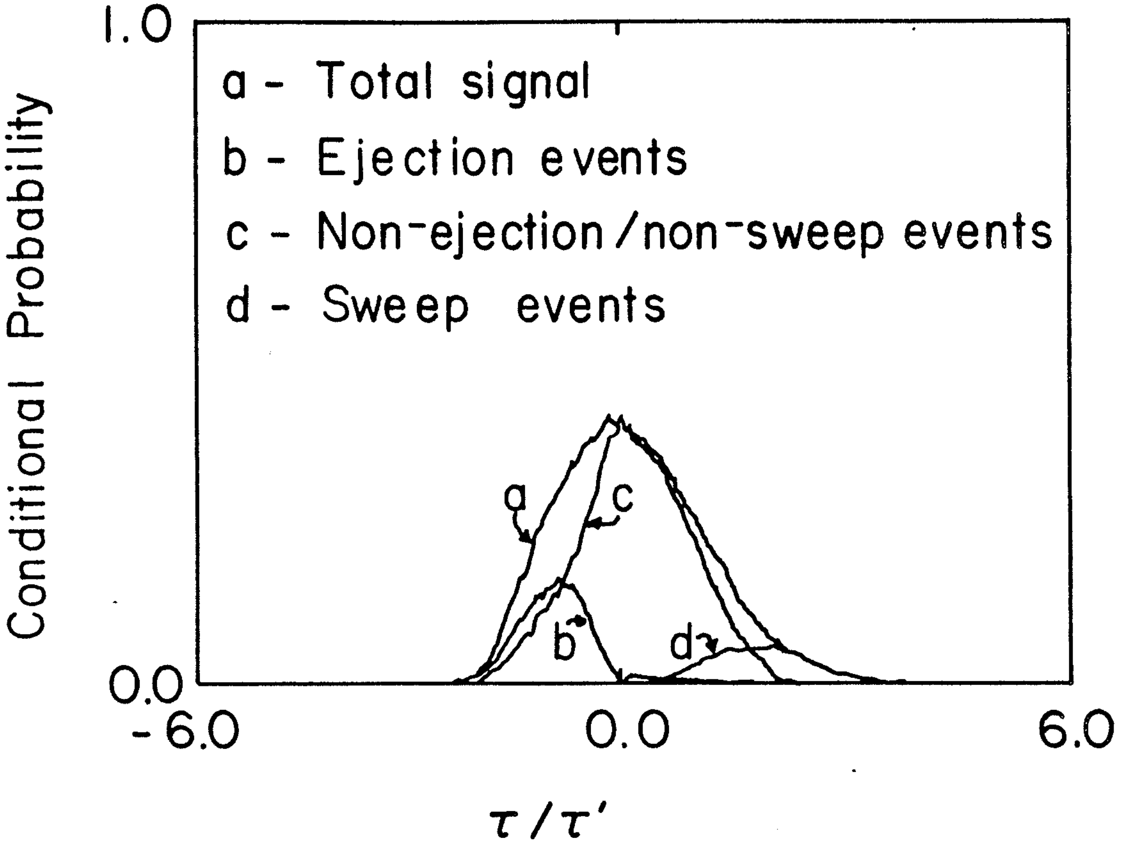


Figure 4.45: The Zaric decomposition of τ at x=39.53 cm.

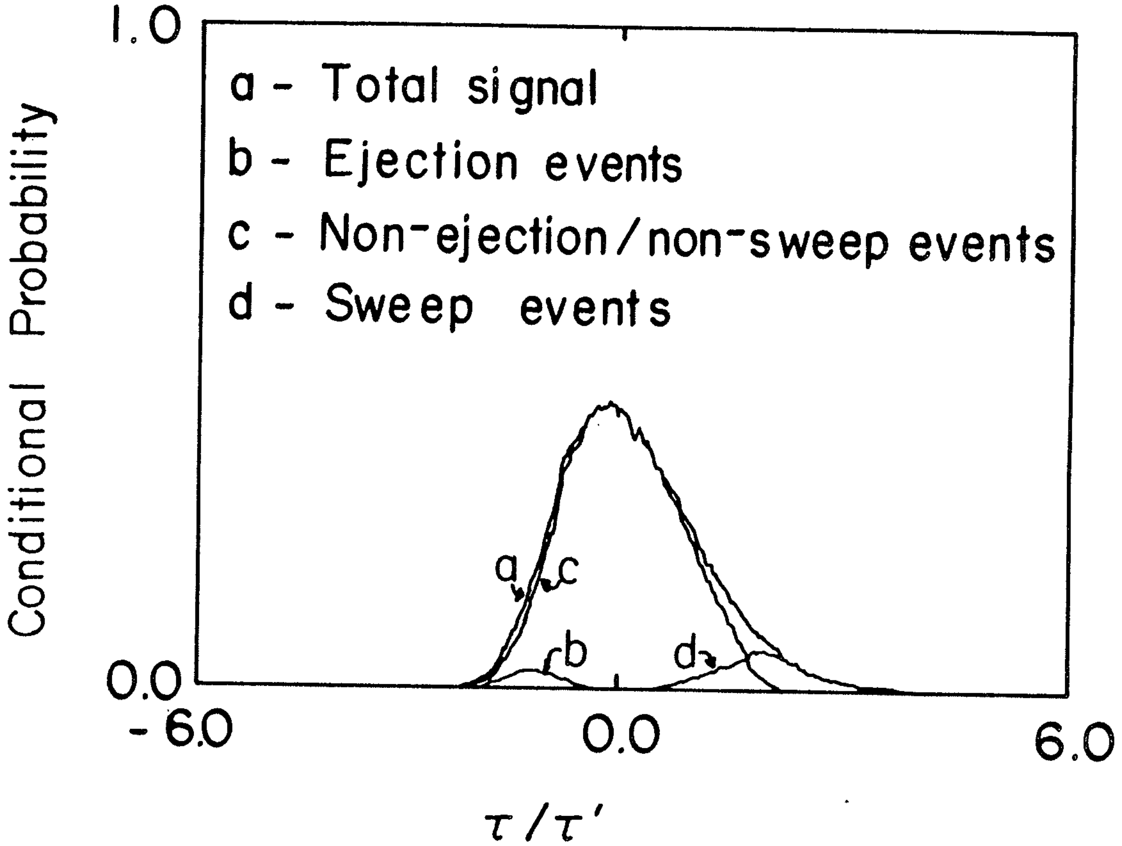


Figure 4.46: The Zaric decomposition of τ at x=50.71 cm.

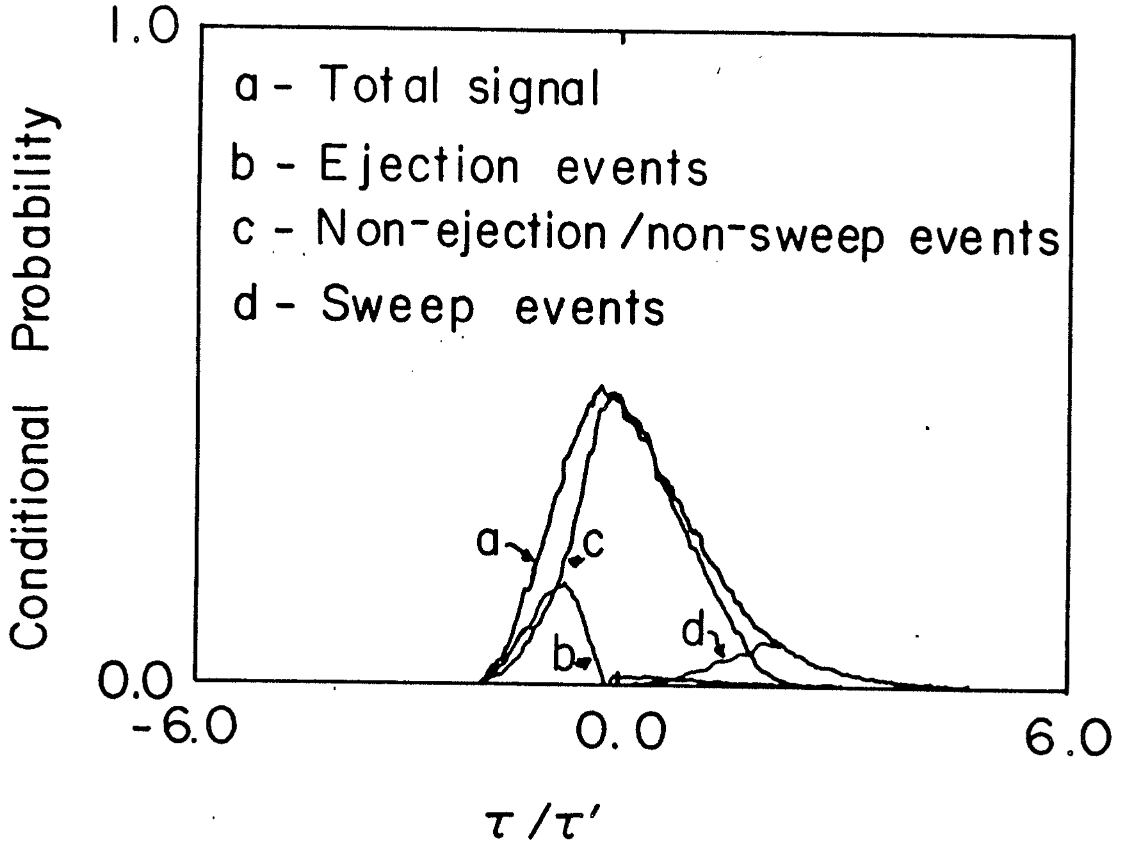


Figure 4.47: The Zaric decomposition of τ at x=55.69 cm.

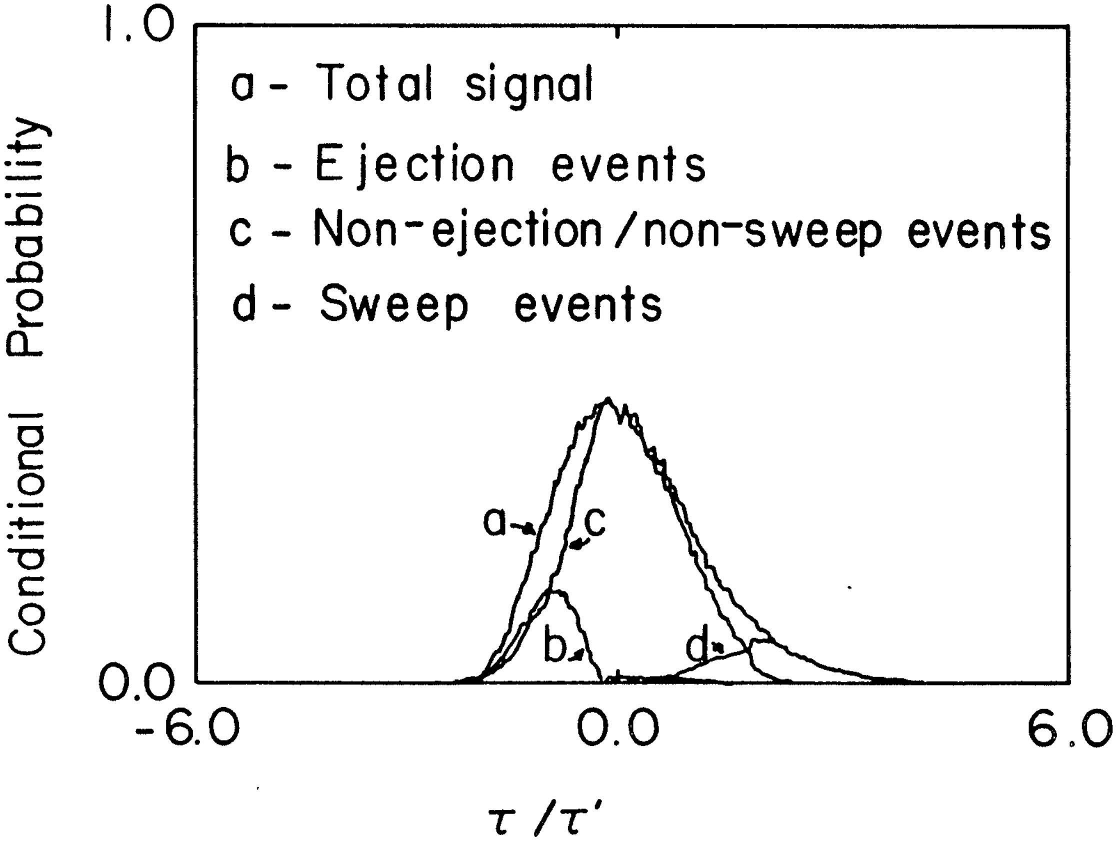


Figure 4.48: The Zaric decomposition of τ at x=69.46 cm.

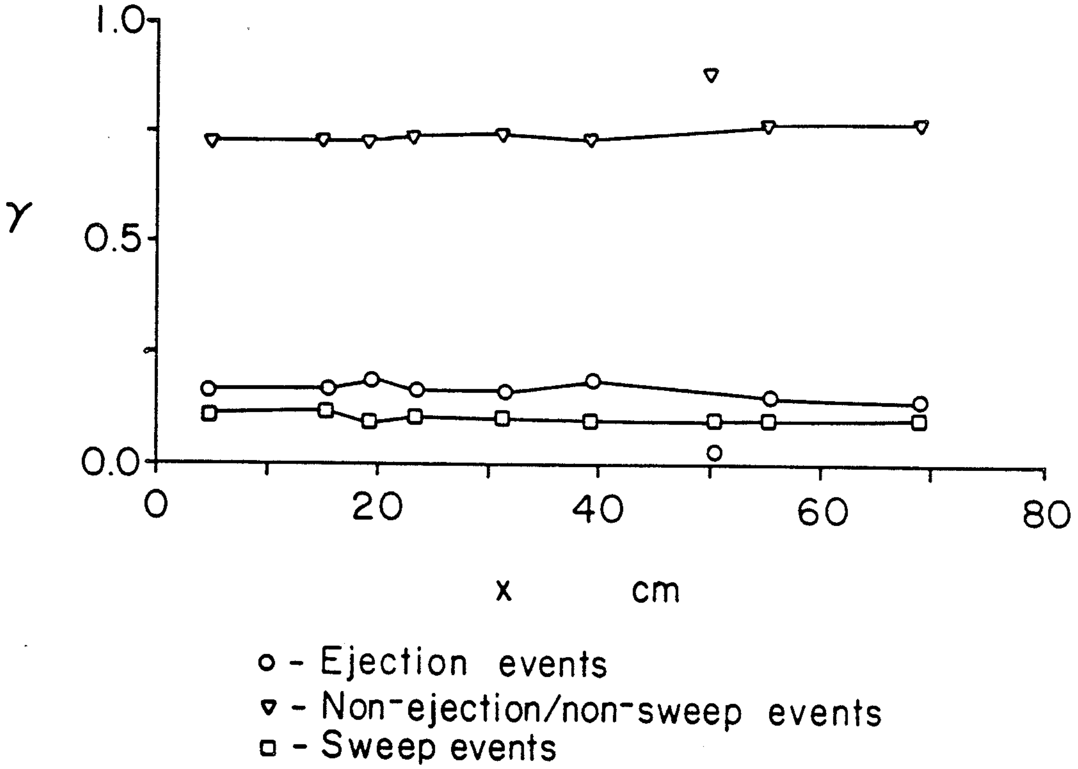


Figure 4.49: The intermittancy of each event in the diffuser.

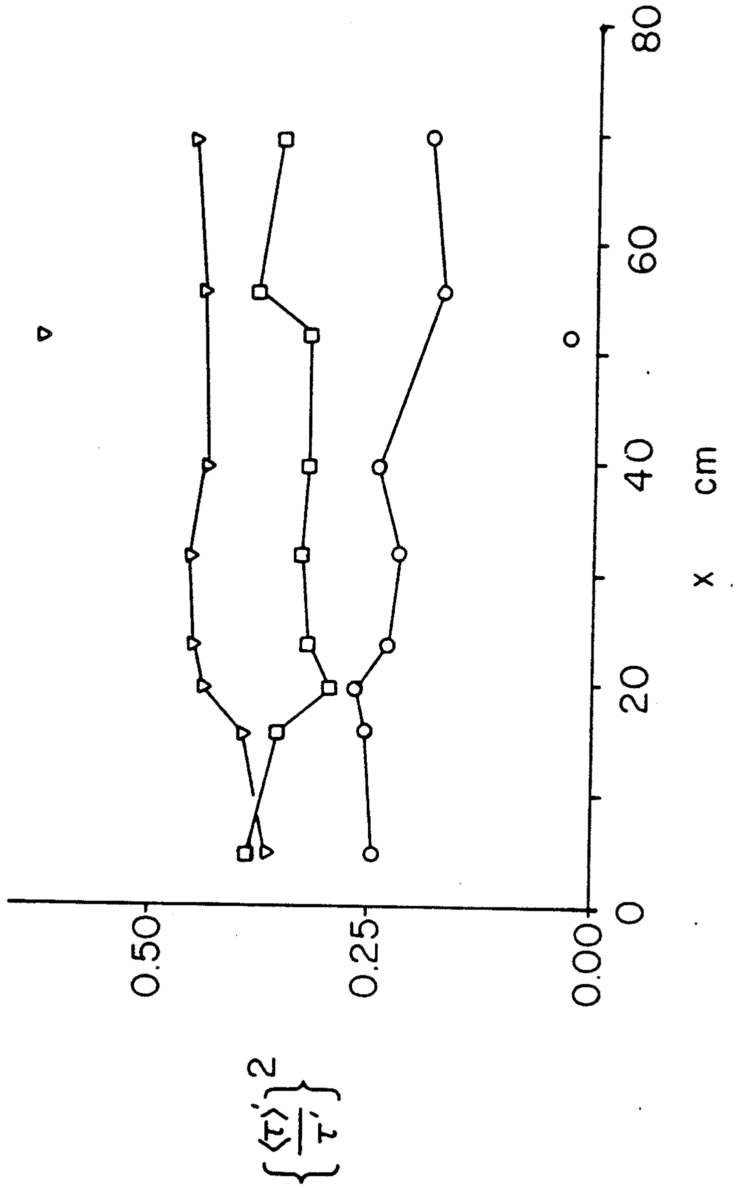


Figure 4.50: The contribution to the mean square by each event. Symbols as in figure 4.49.

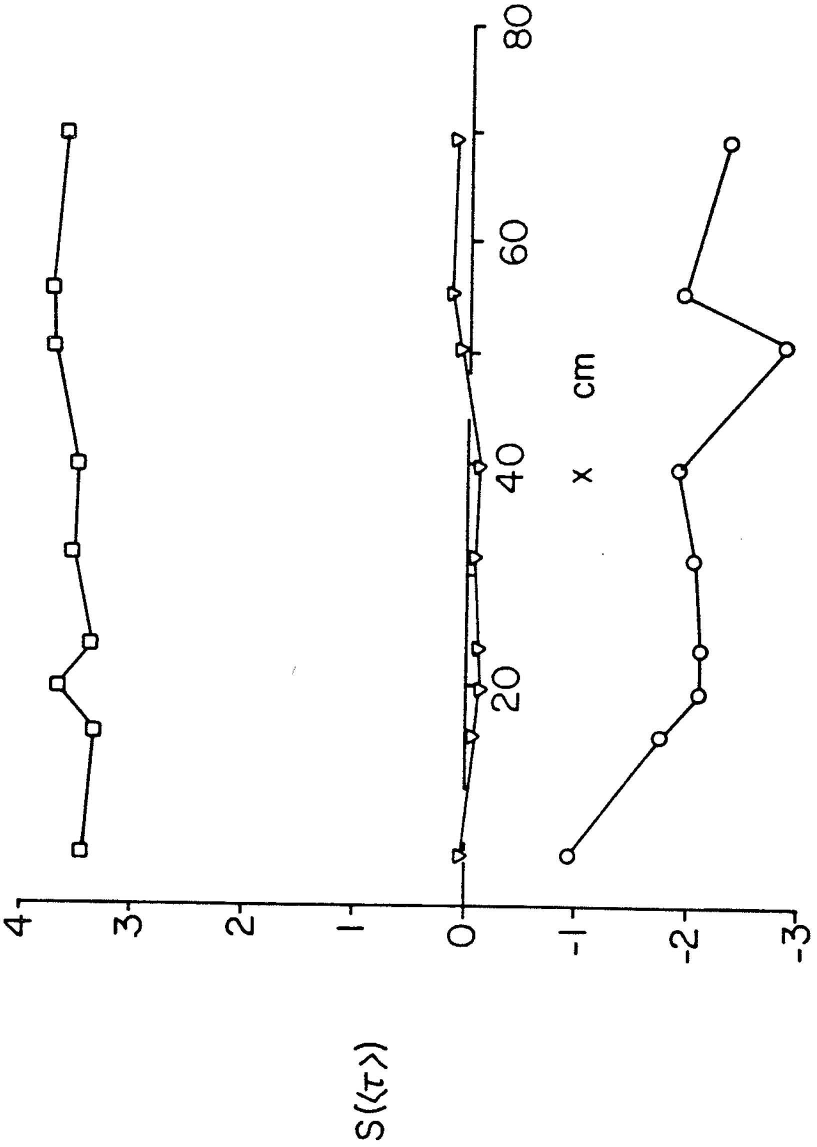


Figure 4.51: The skewness of each event in the diffuser. Symbols as in figure 4.49.

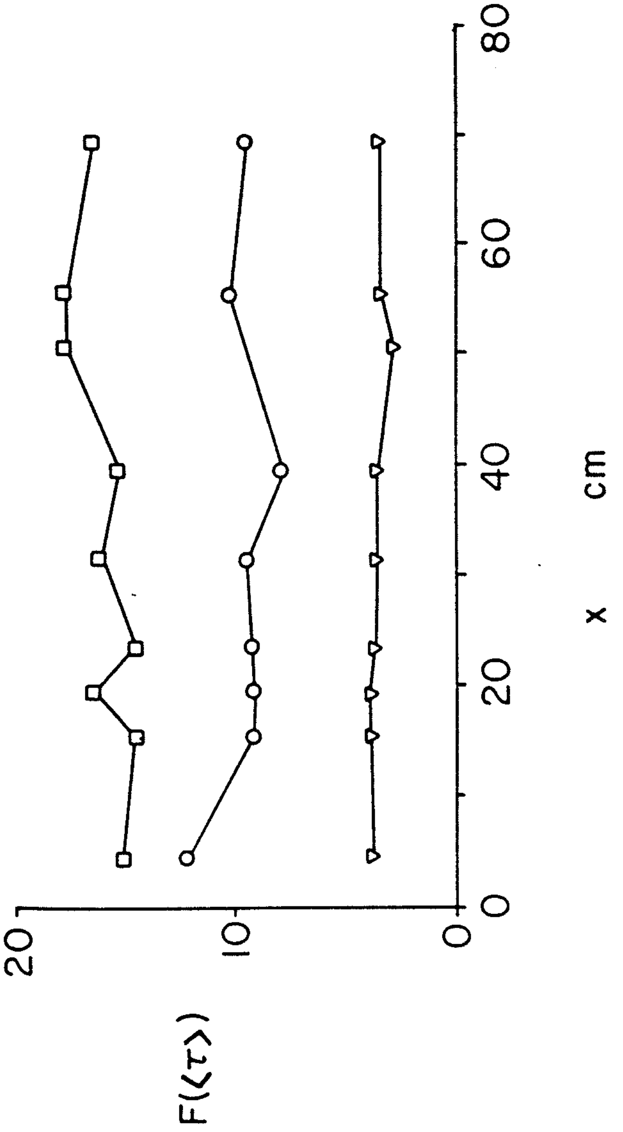


Figure 4.52: The flatness of each event in the diffuser. Symbols as in figure 4.49.

Chapter V

Two Point Measurements

Joint statistical measurements were made of the axial components of the fluctuating wall shear stresses at two points. The measurements were made with a probe fixed at an axial location with a second probe displaced either axially of circumferentially. The measurements were all nondimensionalized by normalizing the τ by τ' at each point. This resulted in the notation $Z_0=\tau_0/\tau'_0$ where the Z are scaled to have a RMS value of 1.

The measurements made were the joint moments up to fourth order, cross-spectra, cross-correlation, coherance and the joint probability density functions. Unfortunately because of the large spacing between the probes the cross-spectra, cross-correlation and coherance became unintelligible in the range of frequencies that could be measured. The bandwidth only extended down to 40 Hz, and the cross-spectra gave only one or two nonzero points above this range. The signal is essentially uncorrelated at frequencies above 40 Hz for the range of spatial separations used in this experiment, see table 2.1. Because of the lack of information in the cross-spectra measurements, they and their derivative quantities will not be discussed further.

5.1 THE JOINT MOMENTS

The joint moments are shown in figures 5.1 through 5.12 inclusive. In general it can be noted that when plotted against Ξ and $\Delta\theta$ the moments appear to fall onto a single curve for each respective moment. This is true for all of the joint moments. The moments were only measured from $x_0=19.58$ cm to the outlet of the diffuser and hence it is not known if this universality holds true near the inlet of the diffuser.

The $\overline{Z_0 Z_1}$ moments, figures 5.1 and 5.7 showed the greatest extent of correlation of all the moment distributions. The 'size' of the largest eddy or structure is bounded to within $-3<\Xi<3$. The correlation for $\Xi<0$ is lower than that for $\Xi>0$, but this asymmetry is small. The $\overline{Z_0 Z_1}$ distribution with $\Delta\theta$ was negative for the smallest $\Delta\theta$. This moment equals one for zero seperation and thus it is obvious many more points for $-45°<\Delta\theta<45°$ are required if a picture of the structure of the shear stresses at a cross-section is desired. The $\overline{Z_0 Z_1}$ moments as a function of $\Delta\theta$ do indicate a behaviour similar to that of the transverse correlation functions of u. They also have negative regions.

The $\overline{Z_0 Z_1^2}$ and $\overline{Z_0^2 Z_1}$ distributions are identical if the curves are reflected about the vertical axis. They also indicate a stronger relationship between the downstream fluctuations than with the upstream fluctuations in τ. The $\overline{Z_0 Z_1^2}$

correlation was nonzero for $-0.5<\Xi<3.0$. The $\overline{Z_0^2 Z_1^2}$ correlation has a minimum value of 1.0 for uncorrelated fluctuations. This distribution was also asymmetric with correlated fluctuations falling to within $0.5<\Xi<0.75$. The $\overline{Z_0^3 Z_1}$ and $\overline{Z_0 Z_1^3}$ moments were symmetric on Ξ, and this moment was nonzero for $-2.5<\Xi<2.5$.

For the joint moments of order greater that two, all the measured points indicated no correspondance of the structure of the flow with $\Delta\theta$. This was due to the overly large initial $\Delta\theta$. This does show that the structure of the flow has a spanwise extent less that D_0. Thus even in a diffuser the large scale structure is larger in its axial than spanwise extent. Hence the mechanisms of turbulence very near the wall are essentially the same as in flat plate and channel flows.

5.2 THE JOINT PROBABILITY DENSITY FUNCTIONS

The joint probability density functions for the shear stresses with x=31.55 cm as the common point are shown in figures 5.13 to 5.20 inclusive, as isometric views. These joint probability density functions are very similar to those with other common points, and hence x=31.55 cm was chosen as a representative data set. The maximum value was approximately 19% (non-dimensionalized) which is larger than that expected from the product of the maximum probability densities from each channel. That value would be approximately 16% (0.4X0.4X100). Visually, all of the joint probability functions look quite similar. The surfaces do fill out more in the positive quadrant as the second point gets nearer to the common point.

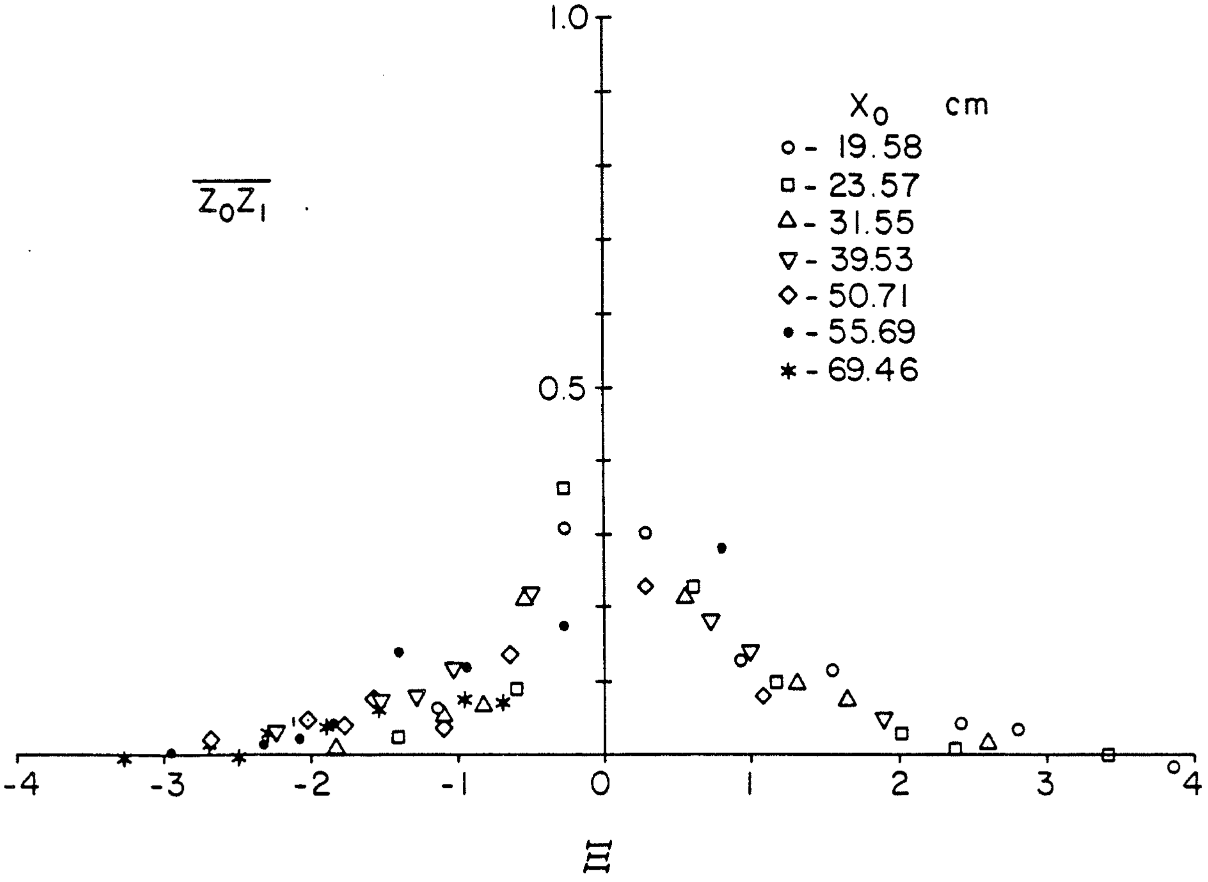


Figure 5.1: The $\overline{Z_0 Z_1}$ moment as a function of Ξ.

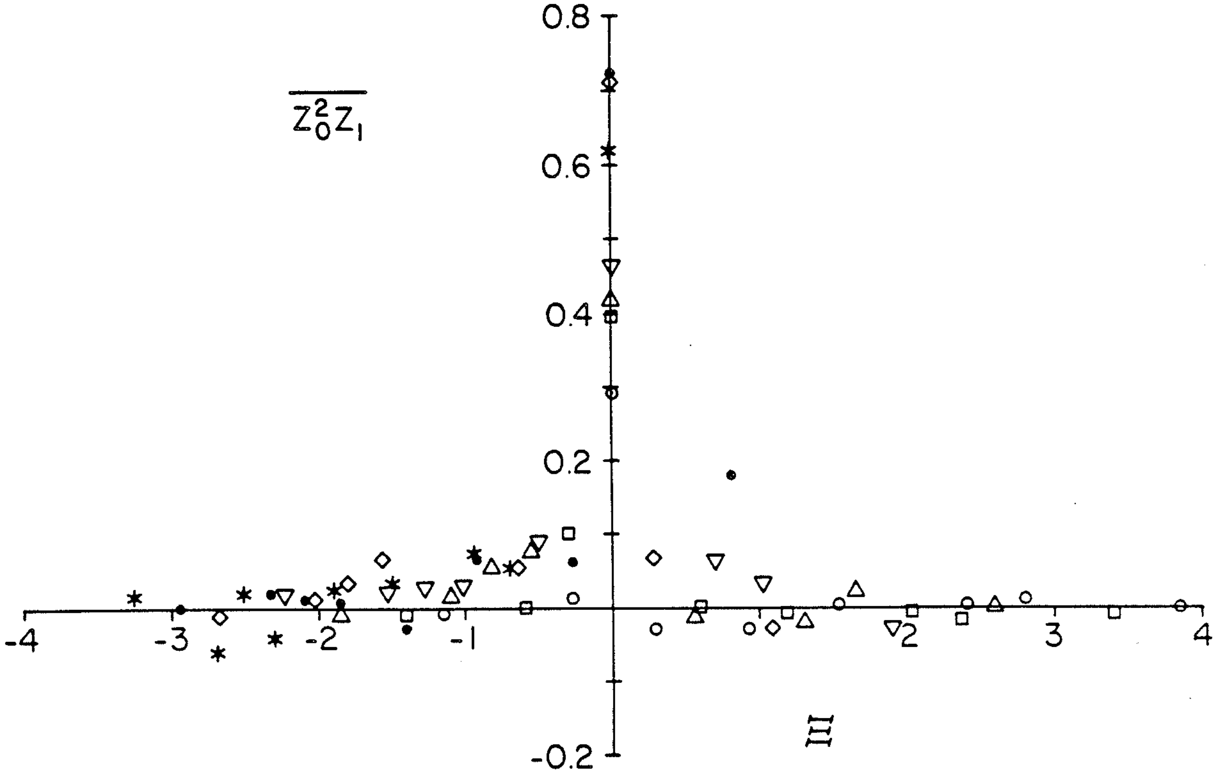


Figure 5.2: The $\overline{Z_0^2 Z_1}$ moment as a function of Ξ. Symbols as in figure 5.1.

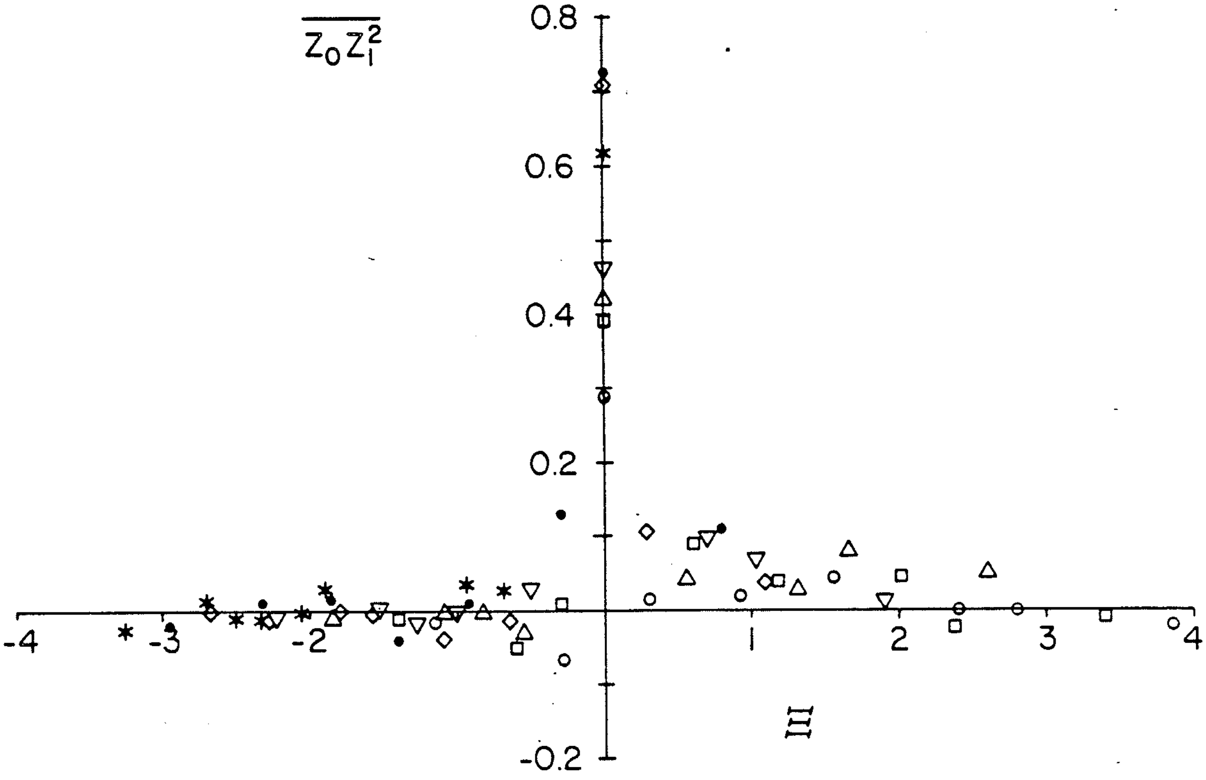


Figure 5.3: The $\overline{Z_0 Z_1^2}$ moment as a function of Ξ. Symbols as in figure 5.1.

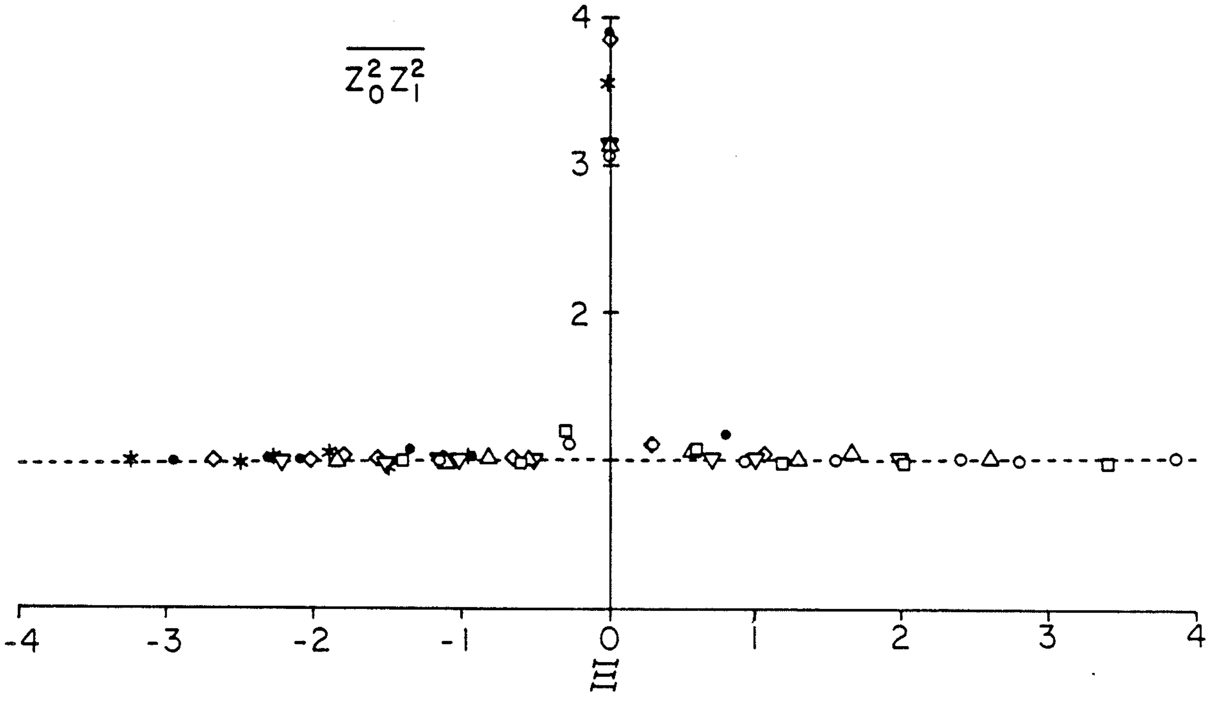


Figure 5.4: The $\overline{Z_0^2 Z_1^2}$ moment as a function of Ξ. Symbols as in figure 5.1.

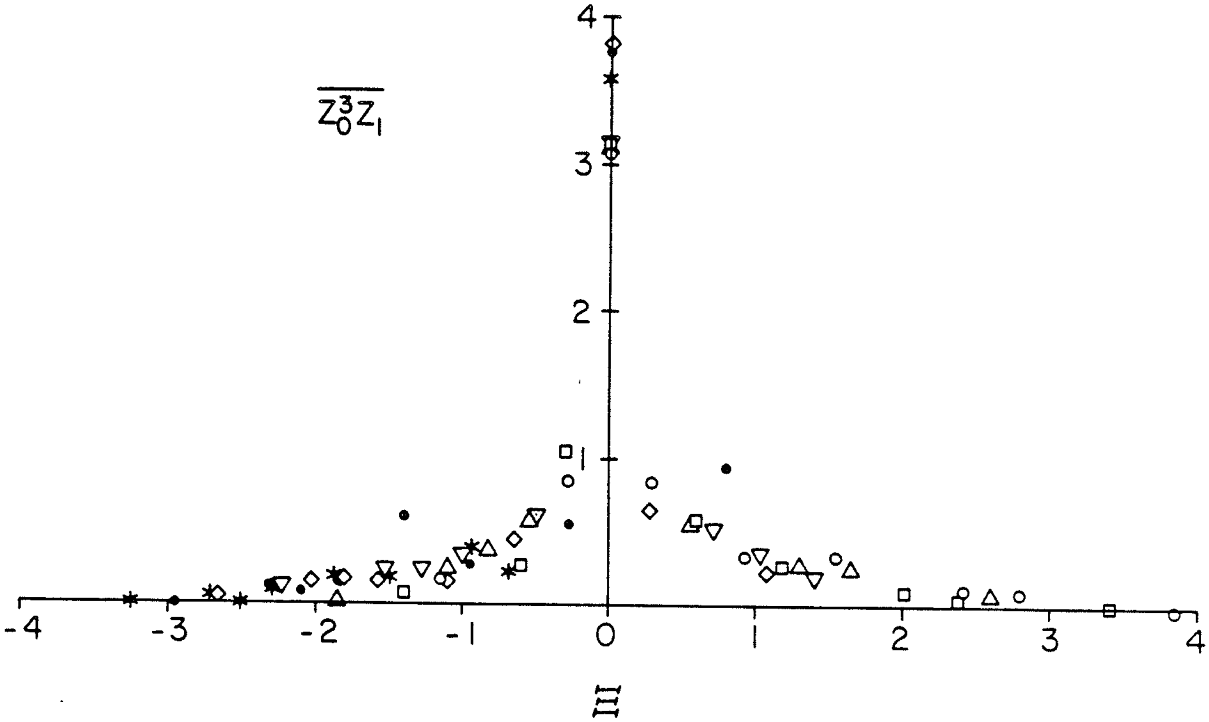


Figure 5.5: The $\overline{Z_0^3 Z_1}$ moment as a function of Ξ. Symbols as in figure 5.1.

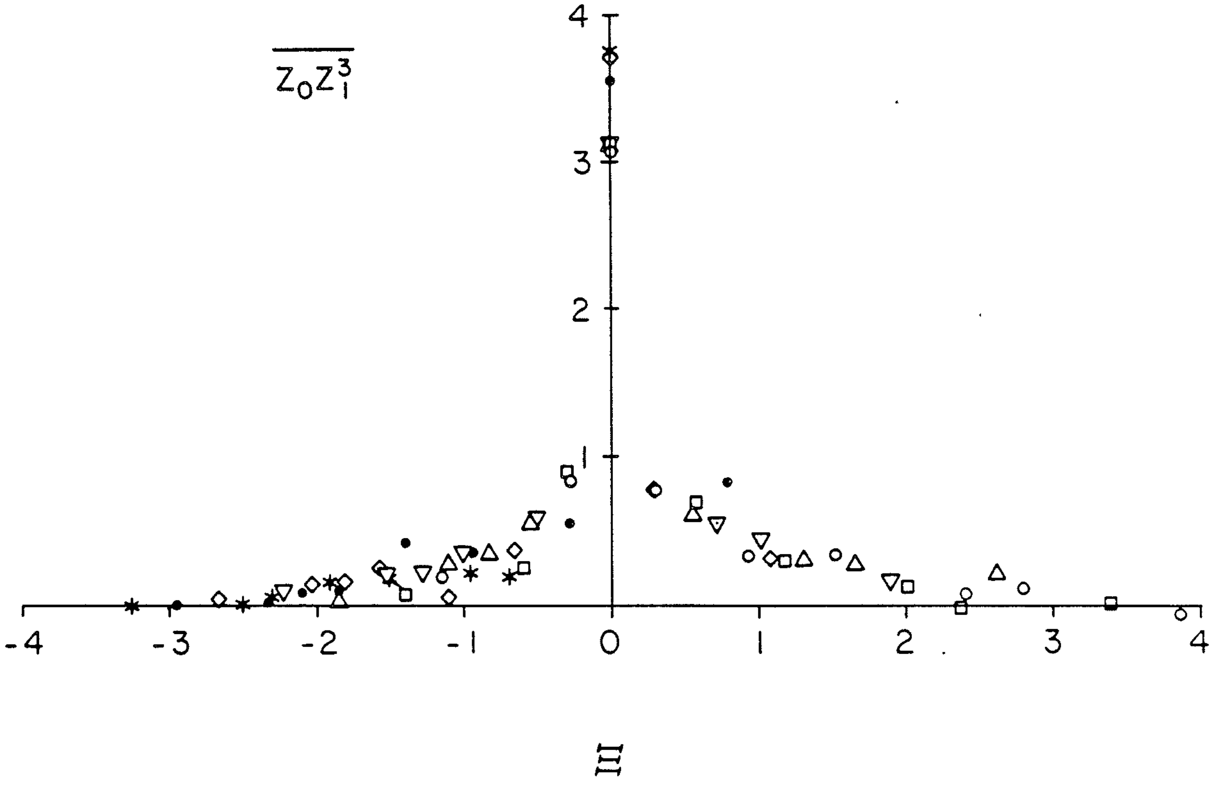


Figure 5.6: The $\overline{Z_0 Z_1^3}$ moment as a function of Ξ. Symbols as in figure 5.1.

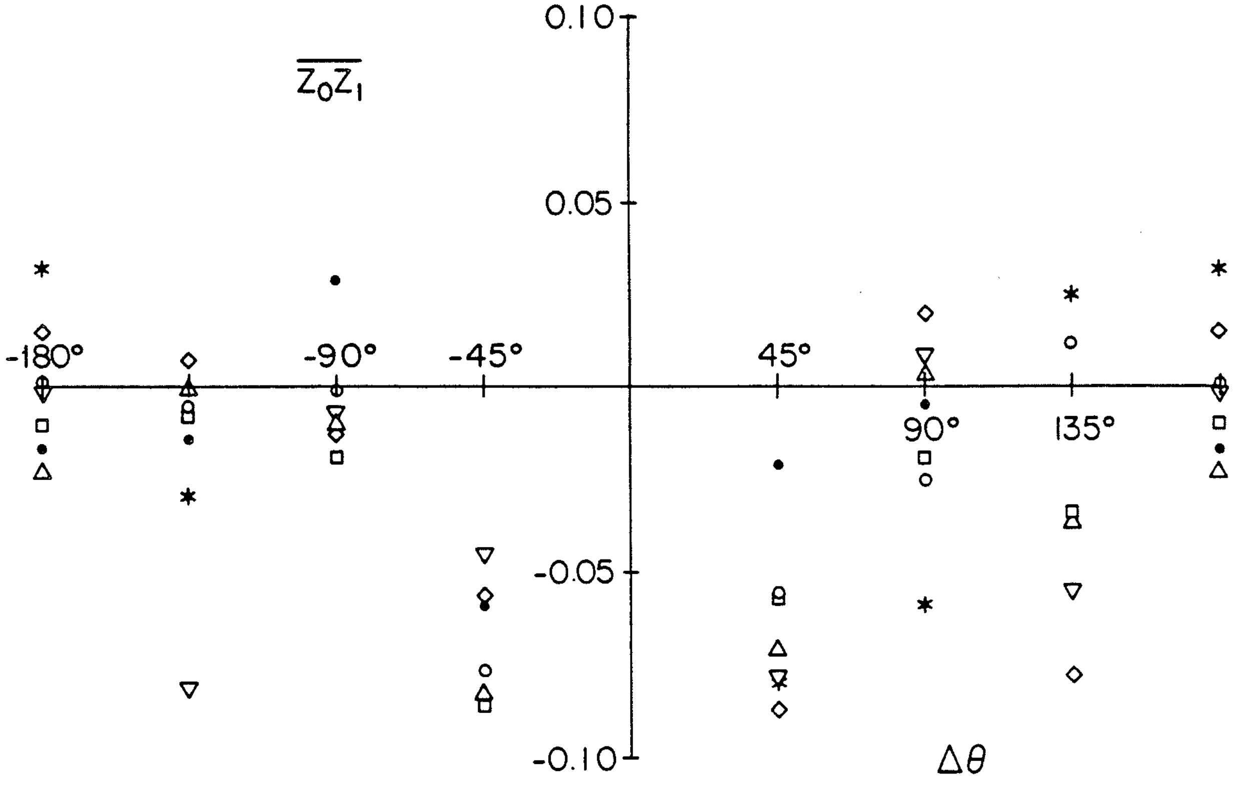


Figure 5.7: The $\overline{Z_0Z_1}$ moment as a function of $\Delta\theta$. Symbols as in figure 5.1.

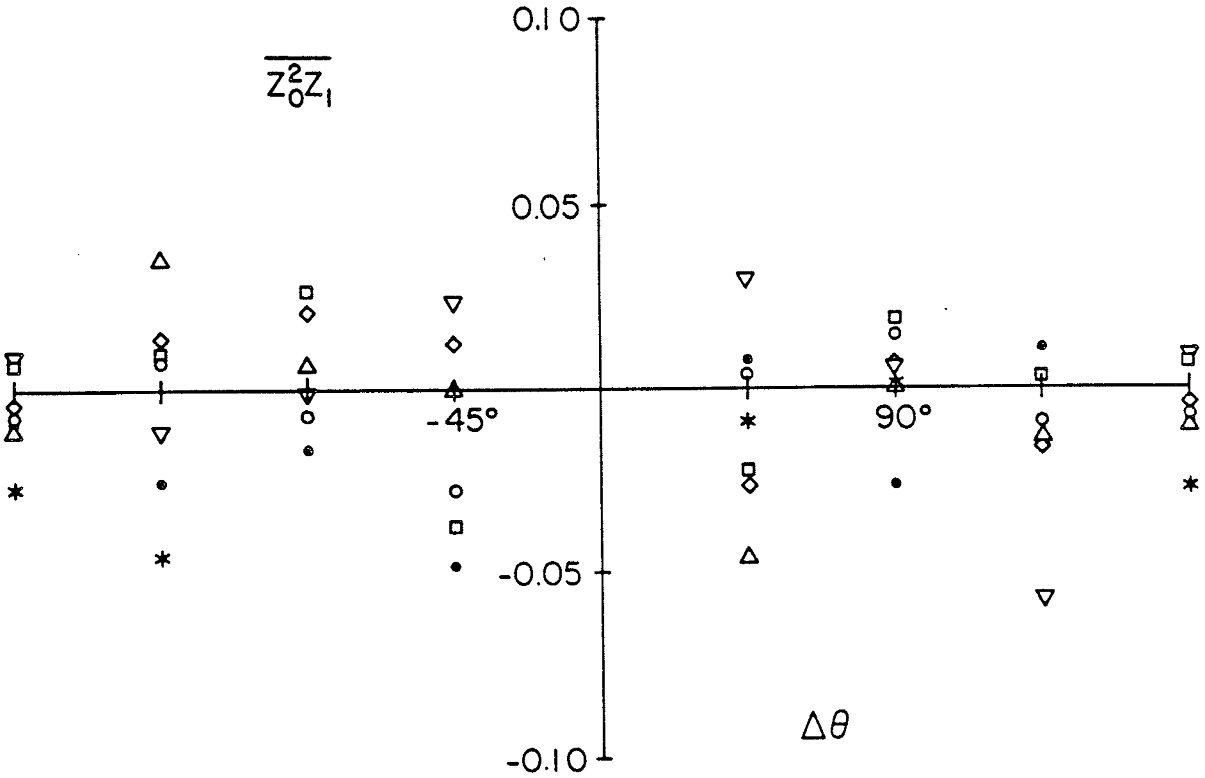


Figure 5.8: The $\overline{Z_0^2Z_1}$ moment as a function of Δθ. Symbols as in figure 5.1.

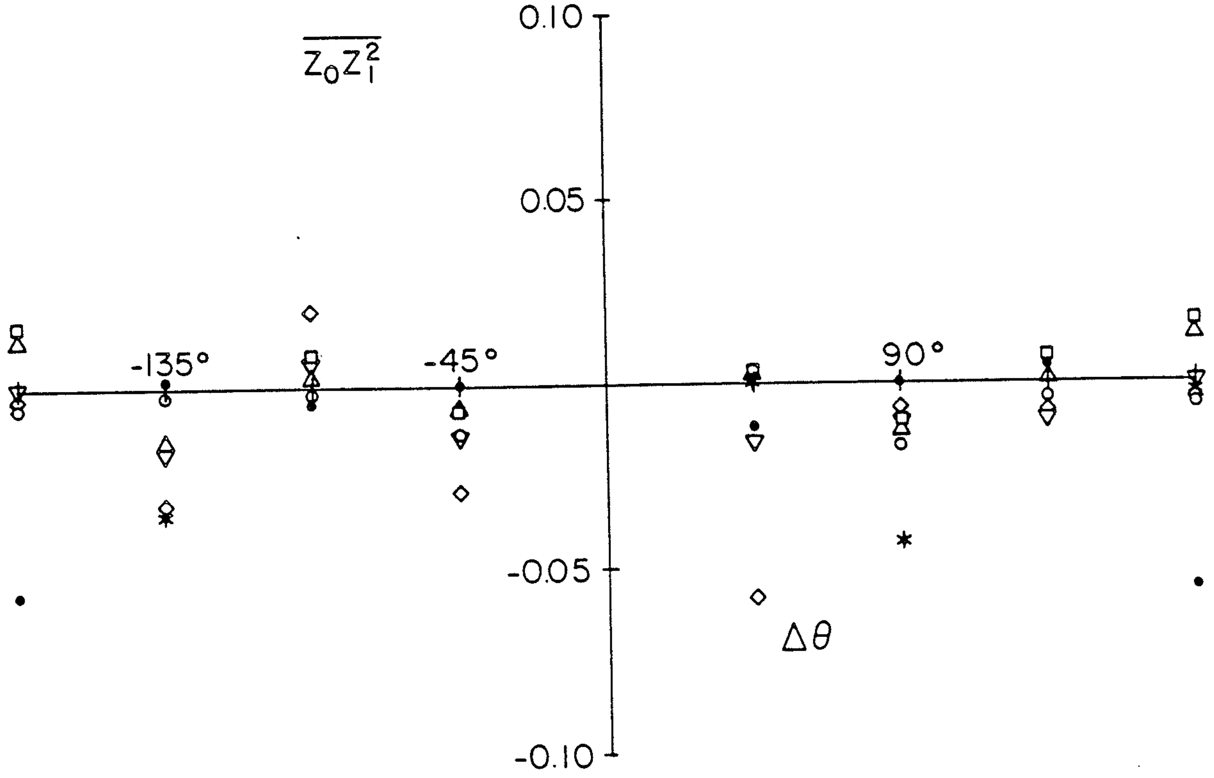


Figure 5.9: The $\overline{Z_0 Z_1^2}$ moment as a function of $\Delta\theta$. Symbols as in figure 5.1.

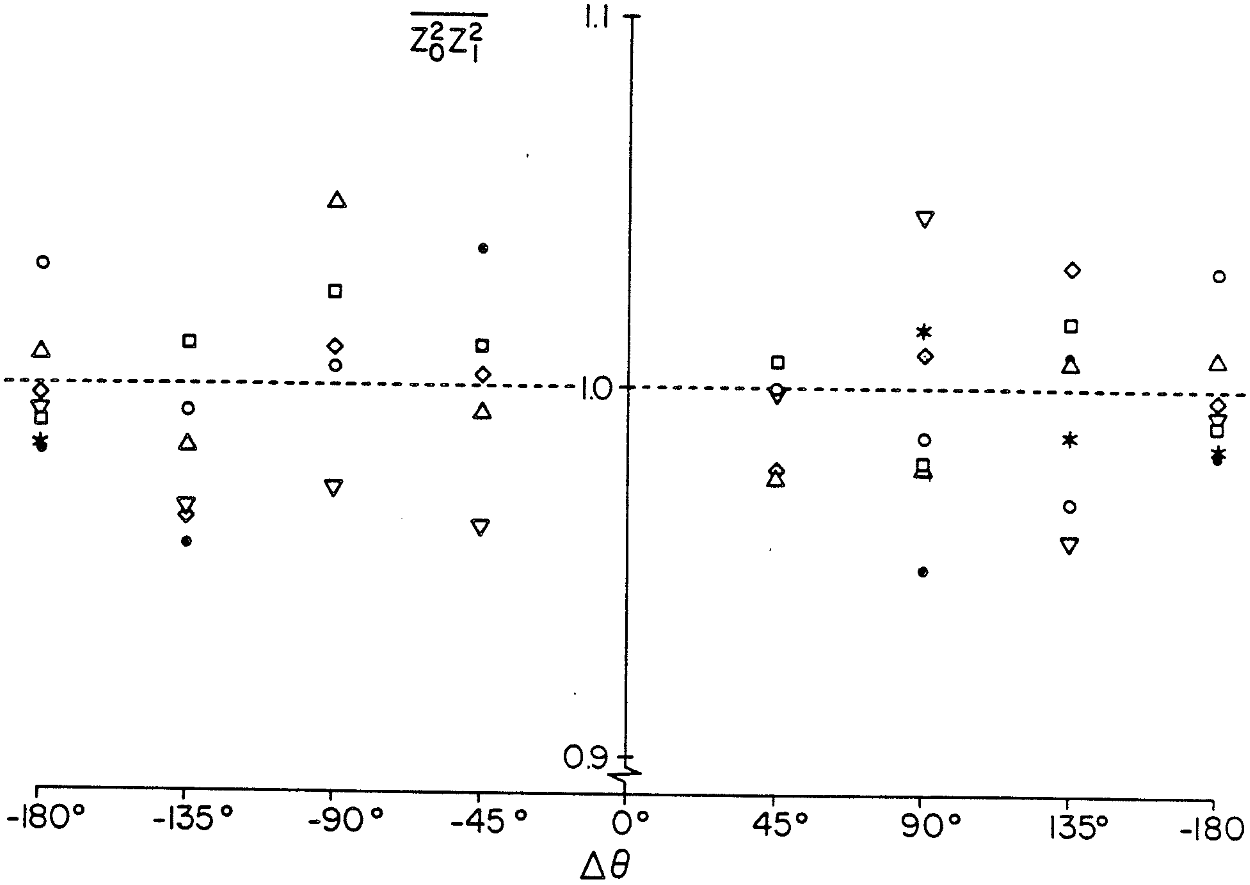


Figure 5.10: The $\overline{Z_0^2 Z_1^2}$ moment as a function of $\Delta\theta$. Symbols as in figure 5.1.

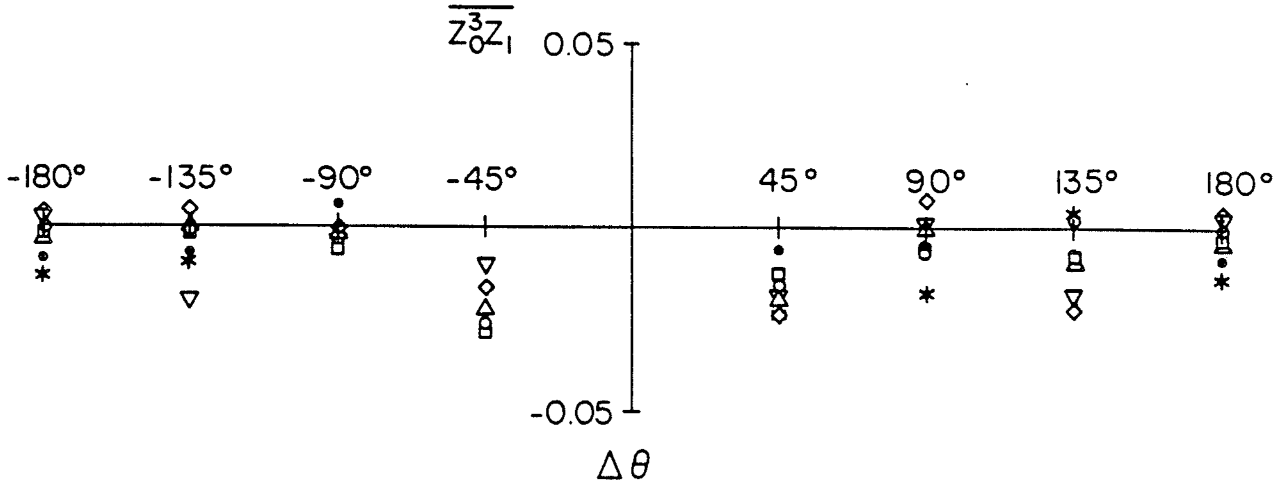


Figure 5.11: The $\overline{Z_0^3 Z_1}$ moment as a function of $\Delta\theta$. Symbols as in figure 5.1.

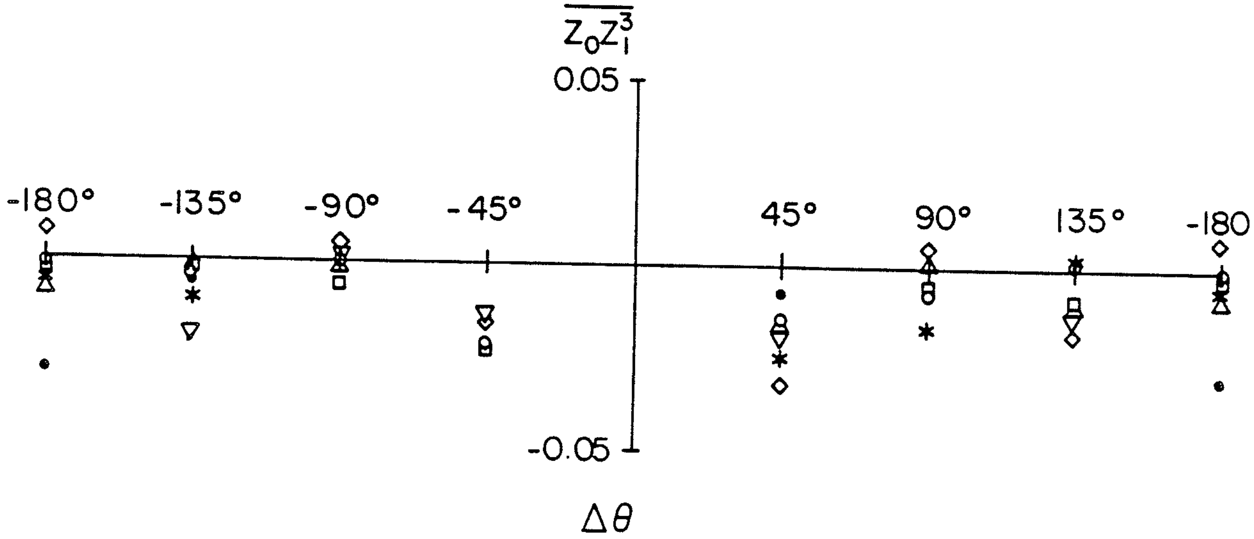


Figure 5.12: The $\overline{Z_0 Z_1^3}$ moment as a function of Δθ. Symbols as in figure 5.1.

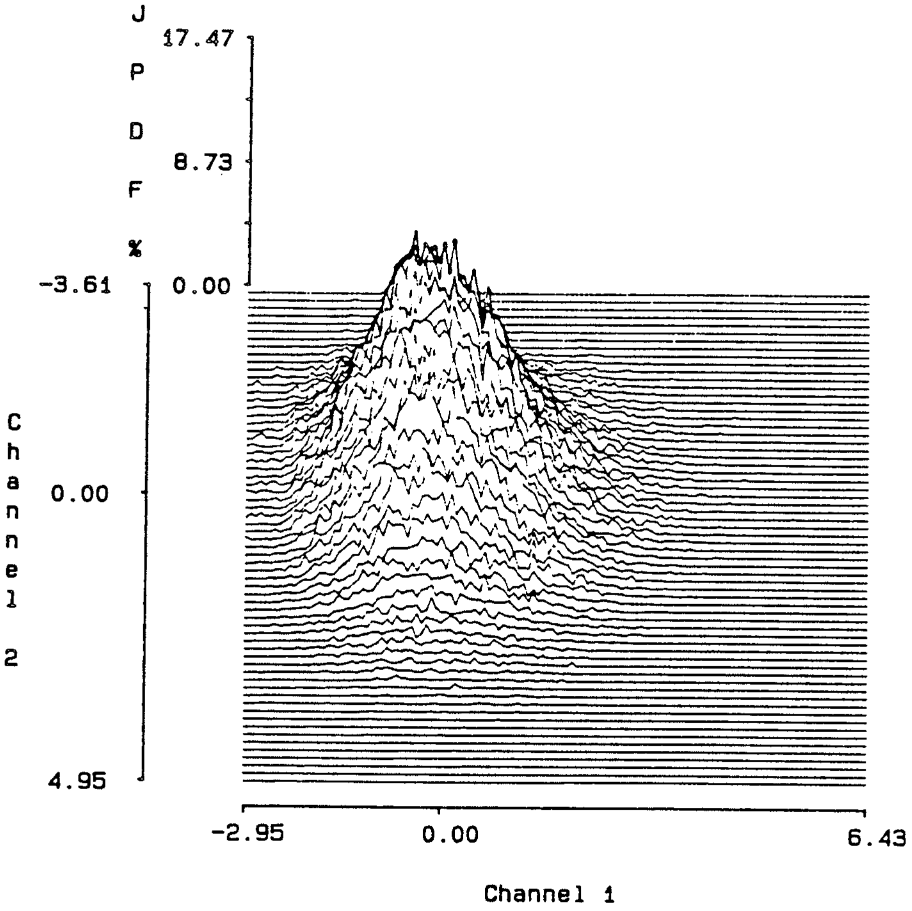


Figure 5.13: The JPDF between x=31.55 cm and x=4.62 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=4.62cm. The signals have been nondimensionalized by their respective RMS values.

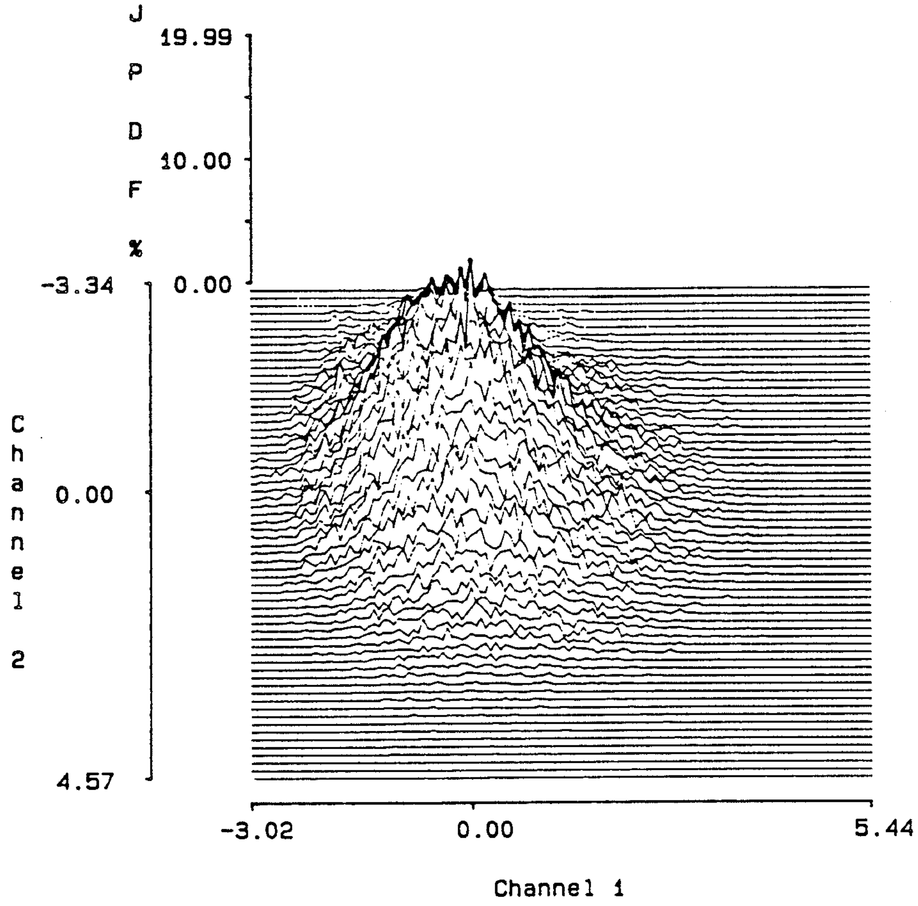


Figure 5.14: The JPDF between x=31.55 cm and x=15.59 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=15.59 cm. The signals have been nondimensionalized by their respective RMS values.

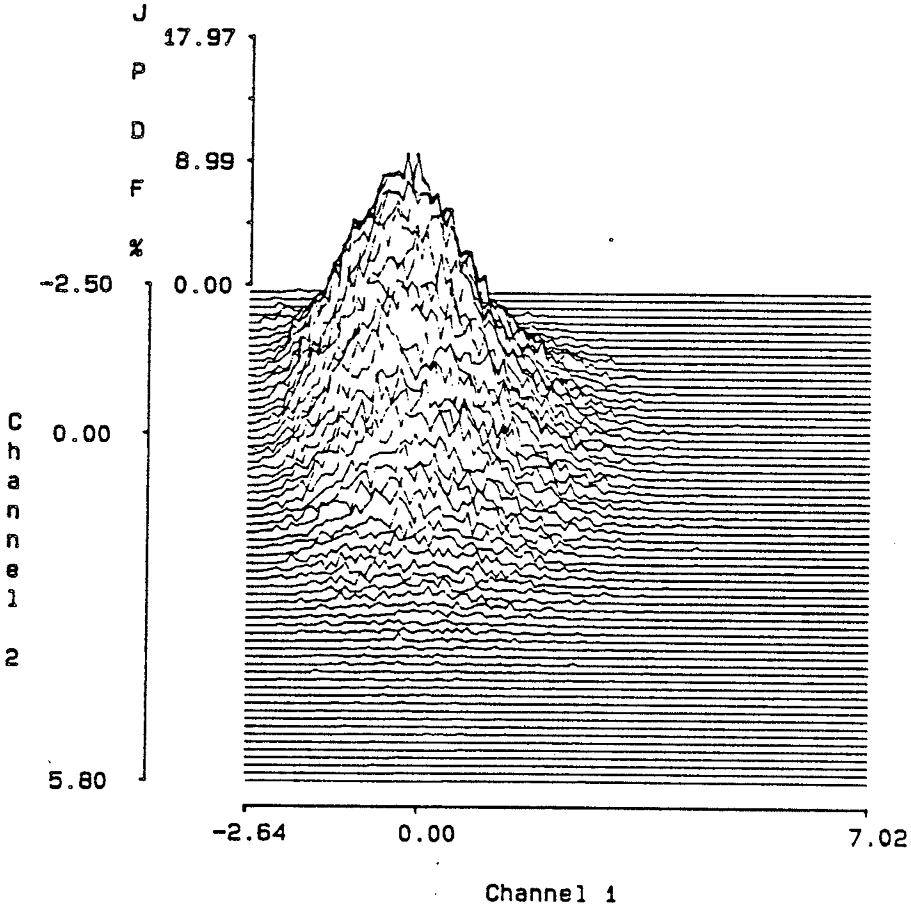


Figure 5.15: The JPDF between x=31.55 cm and x=19.58 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=19.58 cm. The signals have been nondimensionalized by their respective RMS values.

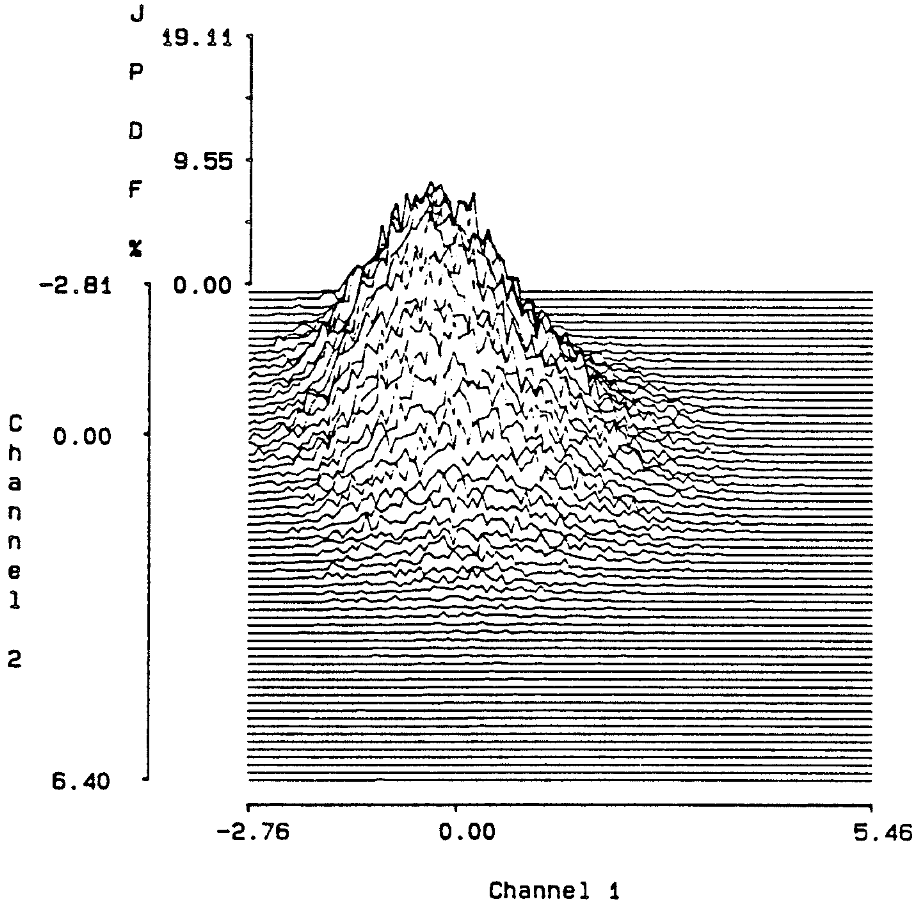


Figure 5.16: The JPDF between x=31.55 cm and x=23.57 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=23.57 cm. The signals have been nondimensionalized by their respective RMS values.

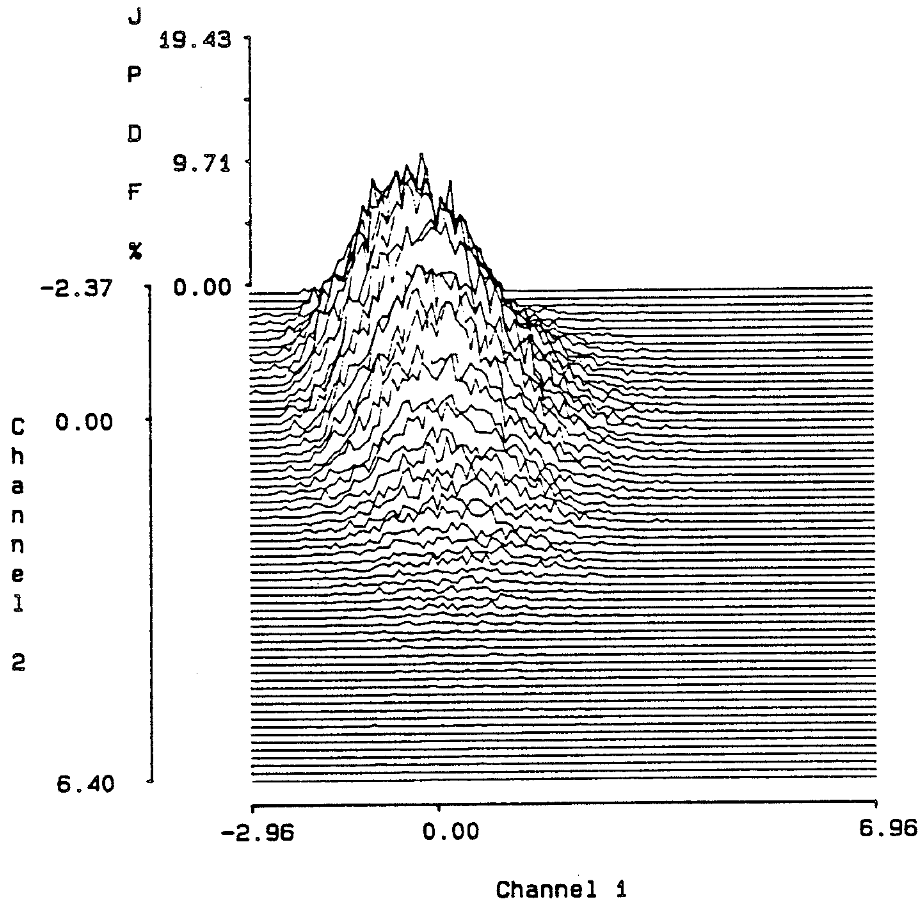


Figure 5.17: The JPDF between x=31.55 cm and x=39.53 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=39.53 cm. The signals have been nondimensionalized by their respective RMS values.

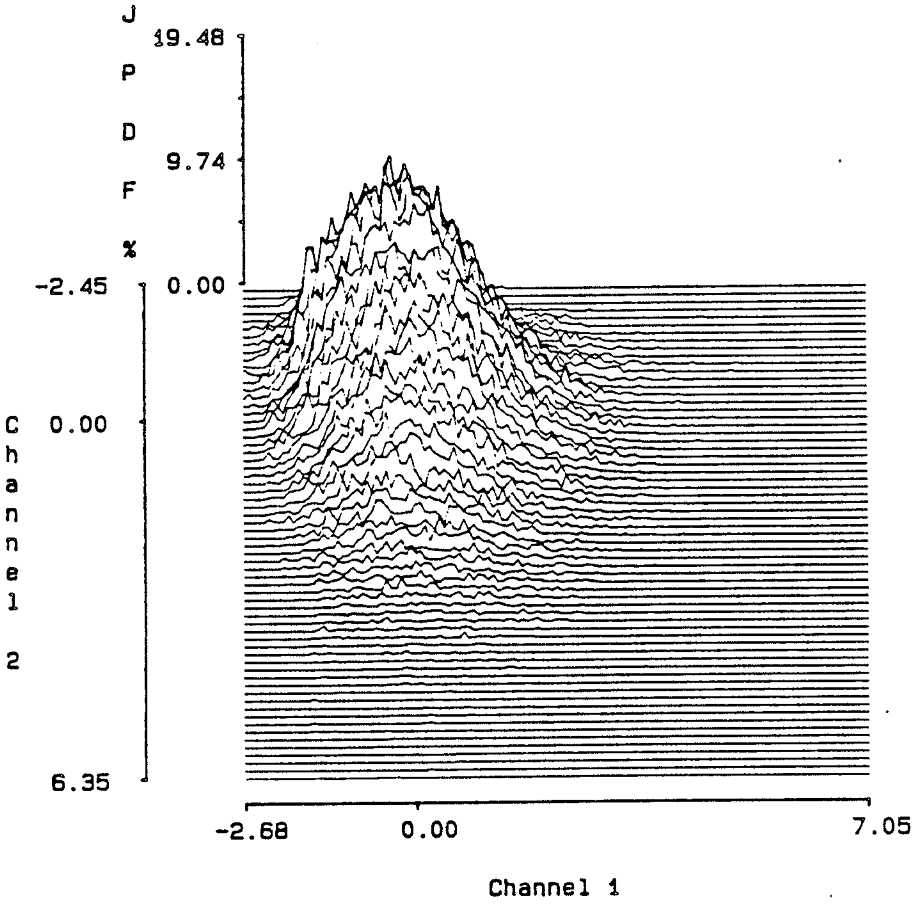


Figure 5.18: The JPDF between x=31.55 cm and x= 50.71 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from 50.71 cm. The signals have been nondimensionalized by their respective RMS values.

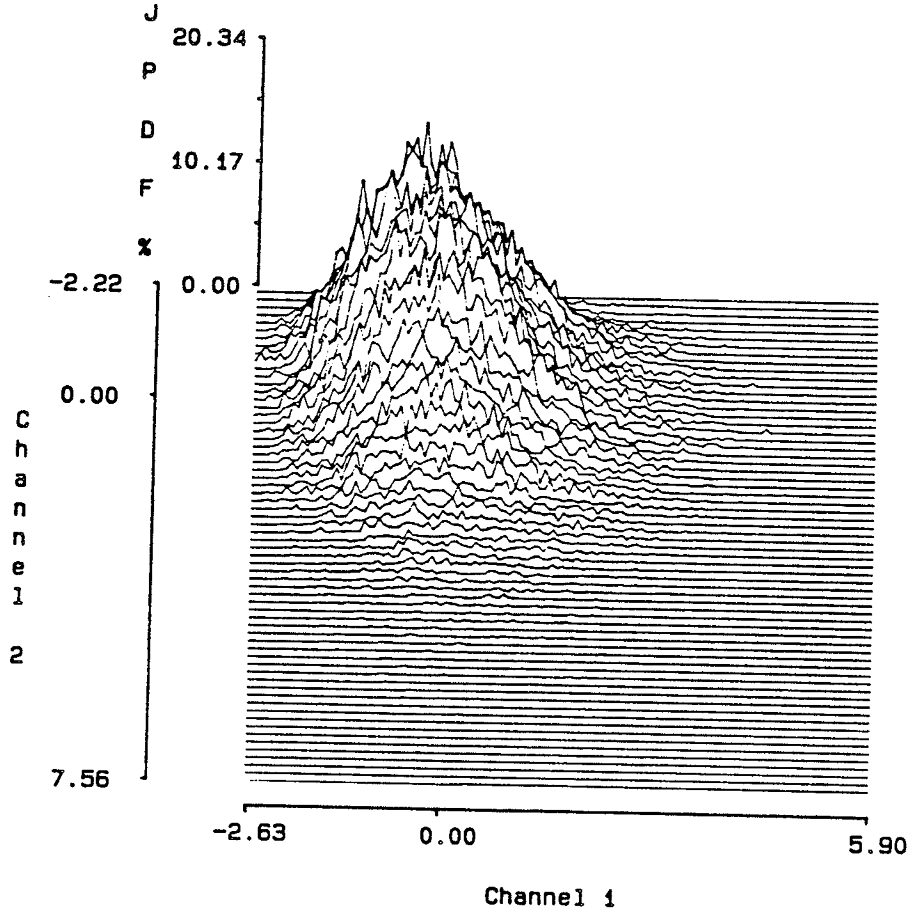


Figure 5.19: The JPDF between x=31.55 cm and x=55.69 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=55.69 cm. The signals have been nondimensionalized by their respective RMS values.

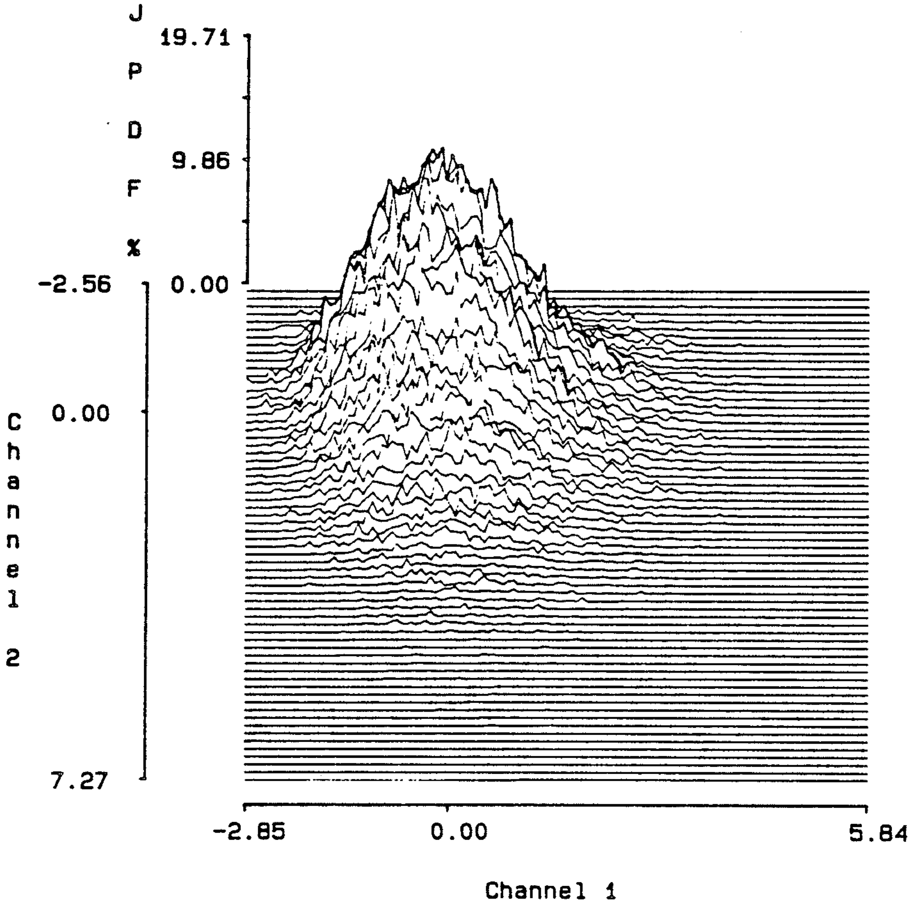


Figure 5.20: The JPDF between x=31.55 cm and x=69.46 cm. Channel 1 was the signal from x=31.55 cm and channel 2 was the signal from x=69.46 cm. The signals have been nondimensionalized by their respective RMS values.

Chapter VI

Conclusions

In conclusion, it may be stated that the nature of the viscous sublayer in the 8° conical diffuser as expressed by fluctuations in the wall shear stress, is structurally similar to the viscous sublayer in flat plate, channel and pipe flows. The increase in the magnitude of the fluctuations of the wall shear stress lagged behind the imposition of the adverse pressure gradient. The central moments of τ were found to vary with axial position however this could be attributed to measurement error. Like Mitchell and Hanratty [1966] the spectra were found to collapse when nondimensionalized by the outer time variable and the RMS of τ.

The probability density functions were also found to be similar when nondimensionalized by τ', which supports the assertion that the viscous sublayer is similar. Zaric's conditional sampling scheme produced nearly identical results as to the probability or intermittancy of ejection and sweep events throughout the diffuser and their relative contribution to the first three central moments.

The two-point joint moments indicated a structure in the flow that was highly elongated in the flow direction. This length was of the order of 5 or 6 local diameters, and the transverse scale was less than one local diameter.

Chapter VII

Recomendations

Any future work along the lines of this study should follow a very detailed examination of the dynamic characteristics of the hot-film probe. This uncertainty in the instrument performance will cloud any hot-film shear stress measurements.

A group of experiments that should yield valuable information would be measuring the wall shear stress-velocity correlations. This should indicate more about the nature of the flow near the wall in an adverse pressure gradient.

References

Arora, S. C. and Azad, R. S., 1980a: Turbulent kinetic energy balance in a flow with adverse pressure gradient, A.I.A.A. paper no. AIAA-80-0073.

Arora, S. C. and Azad, R. S., 1980b: Applicability of the isotropic vorticity theory to an adverse pressure gradient flow, J. Fluid Mech., Vol. 97, part 2, pp. 385-404.

Arora, S. C. and Azad, R. S., 1981: On the moments in an inhomogenious turbulent flow field with adverse pressure gradient, in Flow Its Measurement and Control in Science and Industry, Vol. 2, ed. W. W. Durgin, Instrum. Soc. America, pp. 45-57.

Azad, R. S. and Hummel, R. H., 1979: Similarities of turbulence structure in a conical diffuser with other wall bounded flows, AIAA J.,Vol. 17, No. 8, pp. 884-891.

Azad, R. S. and Hummel, R. H., 1981: Structure functions of turbulence in an adverse pressure gradient, Phys. Fluids, Vol. 24, No. 10, pp. 1774-1779.

Beauchamp, K. G. and Yuen, C. K., 1979: Digital methods for signal analysis, George Allen and Unwin, London.

Bellhouse, B. J. and Schultz, D. L., 1966: Determination of mean and dynamic skin friction, separation and transition in low-speed flow with a thin-film heated element, J. Fluid Mech., Vol. 24, part 2, pp. 379-400.

Bellhouse, B. J. and Schultz, D. L., 1968: The measurement of fluctuating skin friction in air with heated thin-film gauges, J. Fluid Mech., Vol. 32, part 4, pp. 675-680.

Black, T. J., 1966: Some practical applications of a new theory of wall turbulence, in Proc. 1966 Heat Transfer Fluid Mech. Inst., Stanford U. P., Stanford Calif., pp. 366-368.

Black, T. J., 1969: Viscous drag reduction examined in the light of a new model of wall turbulence, in Viscous Drag Reduction, ed. S. C. Wells, Plenum Press, New York, pp. 383-407.

Blinco, P. H. and Simons, D. B., 1974: Characteristics of turbulent boundary shear stress, J. Engineering Mech. Division, Vol. 100, No. EM2, pp. 203-220.

Bradshaw, P., 1965: A compact, null-reading, tilting U-tube micromanometer with a rigid liquid container, J. Sci. Instrum., Vol. 42, pp. 677-680.

Bradshaw, P. and Gregory, N., 1961: The determination of local turbulent skin friction from observations in the viscous sublayer, R.&M. No. 3202.

Brison, J. F., Charnay, G. and Comte-Bellot, G., 1979: Calcul des transferts thermiques entre film chaud et substrat par un modele a deux dimensions: prevision de la response dynamique de sondes usuelles, Int. J. Heat Mass Transfer, Vol. 22, pp. 111-119.

Brown, G. L., 1967: Theory and application of heated films for skin friction measurement, in Proc. 1967 Heat Transfer and Fluid Mech. Inst., Stanford U. P., Stanford Calif., pp. 361-381.

Brown, K. C. and Joubert, P. N., 1969: The measurement of skin friction in turbulent boundary layers with adverse pressure gradients, J. Fluid Mech., Vol. 35, part 4, pp. 737-757.

Chen, C. T., 1979: One-dimensional digital signal processing, Marcel Dekker, New York.

Curel, N., 1962: Laminar boundary layer equations, Oxford Mathematical Monographs, Clarendon Press, London.

Derksen, R. W. and Azad R. S., 1980: Mean strain field in and 8° conical diffuser, Bul. APS, Vol. 25, pp. 1083-1084.

Digital Equipment Corporation, 1976: KW11-K dual programmable real time clock user manual, Digital Equipment Corporation, Maynard Mass., EK-KW11-K-OP-001.

Digital Equipment Corporation, 1977: AD11-K analog to digital converter users manual, Digital Equipment Corporation, Maynard Mass., EK-AD11-OP-002.

Digital Equipment Corporation, 1978: LPA11-K laboratory peripheral accelerator user's guide, Digital Equipment Corporation, Maynard Mass., EK-LPA11-UG-001.

Digital Equipment Corporation, 1979a: LPA11-K FORTRAN user's reference guide, Digital Equipment Corporation, Maynard Mass., AA-H852A-TC.

Digital Equipment Corporation, 1979b: FORTRAN language reference manual, Digital Equipment Corporation, Maynard Mass.,AA-1855D-TC.

Digital Equipment Corporation, 1981a: Peripherals handbook, Digital Equipment Corporation, Maynard Mass., EB-20443-20.

Digital Equipment Corporation, 1981b: IAS/RSX-11 I/O operations reference manual, Digital Equipment Corporation, Maynard Mass., AA-M176A-TC.

Digital Equipment Corporation, 1981c: Laboratory subroutines programmers reference manual, Digital Equipment Corporation, Maynard Mass., AA-C984C-TC.

Digital Equipment Corporation, 1983: RSX-11M/M-PLUS I/O drivers reference manual, Digital Equipment Corporation, Maynard Mass., AA-L677A-TC.

Eckelmann, H., 1974: The structure of the viscous sublayer and the adjacent wall region in a turbulent channel flow, J. Fluid Mech., Vol. 65, part 3, pp. 439-459.

Einstein, H. A. and Li, H., 1956: The viscous sublayer along a smooth boundary, J. Engineer. Mech. Div. A.S.C.E., Vol. 82, EM 2, pp. 945.1-945.27.

Fage, A. and Falkner, V. M., 1931: On the relation between heat transfer and surface friction for laminar flow, R.&M. No. 1408.

Fage, A. and Townend, H. C. H., 1932: An examination of turbulent flow with an ultramicroscope, Proc. Roy. Soc., Vol. A135, pp. 656-677.

Frei, D. and Thomann, H., 1980: Direct measurement of skin friction in a turbulent boundary layer with a strong adverse pressure gradient, J. Fluid Mech., Vol. 101, part 1, pp. 79-95.

Freymuth, P. and Fingerson, L. M., 1977: Electronic testing of frequency response for thermal anemometers, Thermo-Systems Inc. Q., III, Nov., pp.5-9.

Fortuna, G. and Hanratty, T. J.,1971: Frequency response of the boundary layer on wall transfer probes, Int. J. Heat Mass Transfer, Vol. 14, pp. 1499-1507.

Geremia, J. O., 1972: Experiments on the calibration of flush mounted film sensors, DISA Information, No. 13, pp. 5-10.

Goldstein, S., ed., 1938: Modern developments in fluid dynamics, Vol. 1, Clarenden Press, Oxford. (rereleased by Dover Pub., 1965)

Hanratty, T. J., 1956: Turbulent exchange of mass and momentum with a boundary, AIChE. J., Vol. 2, No. 3, pp. 359-362.

Hanratty, T. J., 1967: Study of turbulence close to a solid wall, Phys. of fluids supplement, Vol. 10, pp. s126-s133.

Head, M. R. and Rechenberg, I., 1962: The Preston tube as a means of measuring skin friction, J. Fluid Mech., Vol. 14, pp. 1-17.

Klages, H., 1977: Beitrage zur experimentallen unter suchung und andwendung von heissfilmsonden, Max-Planck-Institute fur Stromungsforshung, Gottingen, Bericht 14.

Kreplin, H. P., 1973: Eine methode sur linearisierung von heisfilmsignalen mit den digitalrechner PDP15 und ihre andwendung bei messungen in einer turbulenten kanalstromung, Max-Planck-Institute fur Stromungsforshung, Gottingen, Bericht 2.

Kreplin, H. P. and Eckelmann, H., 1979: Behaviour of the three fluctuating velocity components in the wall region of a turbulent channel flow, Phys. of Fluids, Vol. 22, pp. 1233-1239.

Landahl, M. T.,1967: A wave-guide model for turbulent shear flow, J. Fluid Mech., Vol. 29, part 3, pp. 441-459.

Landahl, M. T., 1972: Wave mechanics of breakdown, J. Fluid Mech., Vol. 56, part 4, pp. 775-802.

Landahl, M. T., 1975: Wave breakdown and turbulence, SIAM J. Appl. Math., Vol. 28, No.4, pp. 735-756.

Leipmann, H. W. and Skinner, G. T., 1954: Shearing-stress measurements by the use of a heated element, N.A.C.A., Tech. Note 3268.

Lighthill, M. J., 1954: The response of laminar skin friction and heat transfer to fluctuations in the stream velocity, Proc. Roy. Soc., Vol. A224, pp. 1-23.

Ludwieg, H., 1950: Instrument for measuring the wall shearing stress of turbulent boundary layers, N.A.C.A., Tech. Memo. 1284.

Lyatkher, V. M., 1968: Turbulent fluctuations in the viscous sublayer, Soviet Phys. Doklady, Vol. 13, No. 5, pp. 389-391.

Miller, G. E., 1980: Position sensitivity of hot-film shear probes, J. Phys. E: Sci. Instrum., Vol. 13, pp. 973-976.

Mitchell, J. E. and Hanratty, T. J., 1966: A study of turbulence at a wall using an electrochemical wall shear-stress meter, J. Fluid Mech., Vol. 26, part 1, pp.199-221.

Okwuobi, P. A. C. and Azad, R. S., 1973: Turbulence in a conical diffuser with fully developed flow at entry, J. Fluid Mech., Vol. 57, part 3, pp. 602-662.

Patel, V. C., 1965: Calibration of the Preston tube and limitations on its use in pressure gradients, J. Fluid Mech., Vol. 23, part 1, pp.185-208.

Pope, R. J., 1972: Skin-friction measurements in laminar and turbulent flows using heated thin-film gages, AIAA J., Vol. 10, No.6, pp.729-730.

Popovich, A. T. and Hummel, R. L., 1967: Experimental study of the viscous sublayer, AIChE. J., Vol. 13, No. 5, pp. 854-860.

Pozzorini, R., 1976: Das turbulente stromungsteld in einem langen kreiskegel-diffusor, Diss. ETH 5646, Zurich.

Preston, J. H., 1954: The determination of turbulent skin friction by means of Pitot tubes, J. Roy. Aero. Soc., Vol. 58, pp. 109-121.

Py, B., 1973: Etude tridimensionelle de la sous-couche visqueuse dans une veine rectangulaire par des mesures de transfert de matiere en paroi, Internat. J. Heat Mass Transfer, Vol. 16, pp. 129-144.

Py, B. and Duhamel, P., 1972: Mesure des distributions statistiques du gradient parietal de vitesse longitudinale en canal bidimensionnel, C. R. Acad. Sc. Paris, t.275, series A, pp. 373-376.

Ramaprain, B. R. and Tu, S. W., 1983: Calibration of a heated flux gage for skin friction measurement, J. Fluids Engineer., Vol. 105, pp. 455-457.

Reichert, J. K. and Azad, R. S., 1977: Wall shear stress measurement with a hot film in a variable temperature flow, Rev. Sci. Instrum., Vol. 48, No. 3, pp. 341-345.

Sandborn, V. A., 1979: Evaluation of the time dependent surface shear stress in turbulent flows, ASME, paper 79-WA/FE17.

Schlichting, H., 1979: Boundary layer theory, McGraw-Hill Book Co., New York, pg. 597.

Schubert, G. and Corcos, G. M., 1967: The dynamics of turbulence near a wall according to a linear model, J. Fluid Mech., Vol. 29, part 1, pp. 113-135.

Stanton, T. E., Marshall, D. and Bryant, C. N., 1920: On the conditions at the boundary of a fluid in turbulent motion, Proc. Roy. Soc., Vol. A97, pp.413-434.

Sternberg, J., 1962: A theory for the viscous sublayer of a turbulent flow, J. Fluid Mech., Vol. 13, pp. 241-271.

Sternberg, J., 1965: A three-dimensional structure of the viscous sublayer, in Recent Developments in Boundary Layer Research, AGARDograph 97, pp. 1-33.

Sreenivasan, K. R. and Antonia, R. A., 1977: Properties of wall shear stess fluctuations in a turbulent duct flow, J. Applied Mech., Vol. 44(Trans. ASME, series E, Vol. 99), pp.389-395.

van Driest, E. R., 1956: On the turbulent flow near a wall, J. Aeronaut. Sci., Vol. 23, pp. 1007-1011.

van Thinh, N., 1982: A study of the intermittent phases in the wall region of a turbulent flow, in Structure of Turbulence in Heat and Mass Transfer, ed. Z. P. Zaric, Hemispheric P., Washington.

Winter, K. G., 1977: An outline of the techniques available for the measurement of skin friction in turbulent boundary layers, Prog. Aerospace Sci., Vol. 18, pp.1-57.

Zaric, Z.,1972: Analyse statistique de la turbulence pres d'une paroi par echantillonnage conditionnel, C. R. Acad. Paris, t.275, series A, pp. 513-515.

Zaric, Z.,1974: Statistical analysis of wall turbulence phenomena, in Internat. Sym. on Turbulent Diffusion in Environmental Pollution, ed. F. N. Frenkiel and R. E. Munn, Advances in Geophysics, Vol. 18A, pp. 249-261.

Zaric, Z.,1975: Wall turbulence structure and convective heat transfer, Internat. J. Heat Mass Transfer, Vol. 18, pp. 831-842.

Appendix A

The Preston Tube Measurement Technique

The measurement of the shear stress imposed on a solid surface for a turbulent flow is one of the most difficult experimential techniques in fluid mechanics. In this appendix only the method due to Preston [1954] will be discussed. The hot-film method is analyzed in appendix B. For the interested reader shear stress measurements by various methods are described by Winter [1977].

Preston [1954] wanted a simple and yet accurate means of measuring skin friction. He ruled out the use of the momentum equation and measured velocity distributions as tedious and inaccurate. Assuming that a region of the flow near the wall is dependent upon wall variables only (that is $\bar{\tau}$, ρ, ν and some representative length) the shear stress can be uniquely determined. Preston also required that an easily manufactured devise should be used. He proposed that a forward facing impact tube of circular cross-section be used. Through dimensional reasoning it was found that the following functional dependence would hold true

$$P_p d^2/4\rho\nu^2 = f(\bar{\tau} d^2/4\rho\nu^2) \quad . \qquad \ldots \text{A.1}$$

Here the factor 4 is used for convenience and has no real significance. The form of the universial function is to be determined experimentally. Other geometries could be used, with a different calibration, but it was felt that none would be as easy to make as accurately as the Preston tube.

Preston [1954] cautioned that it was necessary that the tubes lie entirely within the region of similarity and that the tube has a cross-sectional area much smaller than any channel it is used in. One should note that there are limitations to the smallest useable diameter due to damping in reading the manometer. Preston made all of his tubes with a constant 0.600 ratio of internal to external diameter as his preliminary experiments showed that this was an important parameter. He also found that one must be careful about burrs.

The calibration given by Preston for smooth or nearly smooth walls with negligible pressure gradients was

$$\log(\bar{\tau} d^2/4\rho\nu^2) = 2.604 + 7/8 \log(P_p d^2/4\rho\nu^2) \quad . \ \ldots \ \text{A.2}$$

Head and Rechenberg [1962] compared measurements of the Preston tube reading against the reading of a sublayer fence and a Stanton tube for both developed pipe flow and boundary layer flow. This comparison was made over a broad range of wall shear stresses. They felt that their results vindicated Preston's method. There had been some objection to the concept of a universal region of wall similarity prior to this, as some measurements of the shear stress by the Preston tube underestimated the value obtained by other methods by about 14%. Using the same experimental rig as Preston, Head and Rechenberg found that there were significant circumferential variations in skin friction or wall shear stress. They attributed this to the earlier inconsistencies in indicated skin friction. As well, Head and Rechenberg examined the effect of an adverse pressure gradient on the Preston tube. They found that on comparison to a sublayer fence the Preston tube method was in excellent agreement except very close to separation.

Patel [1965] reexamined Preston's original calibration as well as experiments in strong favourable and adverse pressure gradients. Patel's results indicated a universal calibration, but of a slightly different form than that given by Preston. It was also found that the ratio of inner to outer diameter had a negligible effect on the calibration. His calibration is given in three different ranges as

Region 1 : $y^* < 1.5$; $u_* d/\nu < 11.2$

$$y^* = x^*/2 + 0.037 \qquad \ldots \text{A.3}$$

Region 2 : $1.5 < y^* < 3.5$; $11.2 < u_* d/\nu < 110$

$$y^* = 0.8287 - 0.1381x^* + 0.1437x^{*2} - 0.0060x^{*3} \qquad \ldots \text{A.4}$$

Region 3 : $3.5 < y^* < 5.3$; $110 < u_* d/\nu < 1600$

$$x^* = y^* + 2\log(1.95y^* + 4.10) \qquad \ldots \text{A.5}$$

where $y^* = \log(\bar{\tau} d^2/4\rho\nu^2)$ and $x^* = \log(P_p d^2/4\rho\nu^2)$. In both favorable and adverse pressure gradient the Preston tube tended to overestimate the skin friction. It was found that increasing errors were registered with increasing Preston tube diameter. The limits that Patel placed on the accuracy were: for adverse pressure gradients - maximum error 3% : $0<\Delta<0.01$, $u_* d/\nu <200$ - maximum error 6% : $0<\Delta<0.015$, $u_* d/\nu<250$ and for favorable pressure gradients - maximum error 3% : $0>\Delta>-0.005$, $u_* d/ <200$ - maximum error 6% : $0>\Delta>-0.007$, $u_* d/\nu<200$, with $d\Delta/dx< 0$. Here $\Delta=(\nu/\rho u_*^3)dP/dx$ is the pressure gradient parameter. Generally Patel's calibration is considered the definitive calibration and tradition will be carried on here.

Later Brown and Joubert [1969] specifically examined the measurement in adverse pressure gradients. They used Patel's calibration but found that the calibration could be used in much stronger adverse pressure gradients than previously indicated by Patel. Brown and Joubert gave the following limits on the accuracy of the Preston tube:

maximum error (%)	$(dP/dx)d/u_*^2$
1	1.34
2	1.74
3	2.06
5	2.55
7	2.98

provided that $\Delta<0.05$.

A very exhaustive examination was done by Frei and Thomann [1980], for strong adverse pressure gradients. They found the Preston tube error to depend upon both Δ and u_*d/ν but not on upstream history. Their results corresponded very well with Patel's calibration for zero pressure gradient. Frei and Thomann were the first to provide a means of correcting the Preston tube results for adverse presure gradient. They found

$$100(\bar{\tau}_m-\bar{\tau})/\bar{\tau} = 9.68\Delta^{1.337}(u_*d/\nu)^{0.857} \qquad \ldots \text{A.6}$$

correlated the deviation between the skin friction indicated by the Preston tube, $\bar{\tau}_m$, and the actual skin friction. This form is inconvenient to use as u_* is not known until after the correction has been employed. Thus Frei and Thomann provided tables so the uncorrected measurements could be directly used to estimate the error. Equation A.6 could be used to iteratively solve for the correction.

In conclusion it appears that the Preston tube can be used to accurately estimate the skin friction in zero pressure gradient and adverse pressure gradient flows. As well the calibration given by Patel used in conjunction with Frei and Thomann's correction should be used.

Appendix B

The Hot-film Measurement Technique

The hot-film method of measurement of wall shear stress is based on the assumption that a unique relationship exists between the rate of heat transfered from a heated element submerged in a surface and the skin friction on the surface. A considerable effort has been made by various researchers to develop and test this instrument. It is the objective of this appendix to highlight these efforts and provide an outline for the use of hot-film shear stress gauges.

Fage and Falkner [1931] presented some very interesting work on the relationship between heat transfer and surface friction for laminar flow. They obtained the relation where the heat transfer is proportional to the 1/3 power of the skin friction. In addition to this, they considered the effect of substrate heating and turbulence in the fluid stream. Experimentally they found that the substrate acted as a heat sink that dissipated a fixed amount of heat, at a given temperature elevation, and that this heat dissipation added a fixed amount to the indicated heat transfer. The substrate heating upstream of the element was found to influence the amount of heat transfered through the constant of proportionality. Downstream from the element they concluded that the substrate heating had an insignificant influence on the rate of heat transfer. They also found that fairly strong favourable pressure gradients, on the front of a circular cylinder, did not appreciably influence the heat transfer relationship. Although it is uncertain from their report whether the constant of proportionality is the same, Fage and Falkner found that the 1/3 relationship held for the turbulent flows they examined. Unfortunately this work was only for steady flow conditions and does not indicate whether a dynamical relationship exists.

Ludwieg [1950] worked on deriving the heat transfer relationship and calibration of the heated element shear stress gauge for turbulent flows. He showed that when the thermal boundary layer remains within the viscous sublayer the Nusselt number or heat transfer coefficient was independent of the Prandtl number and proportional to the third root of the wall shear stress. This is similar to what Fage and Falkner found for laminar boundary layers. Ludwieg also argued that the viscous sublayer was virtually unaffected by pressure gradients and thus that the hot-film shear stress gauge would also be unaffected by the pressure gradients. It would register the skin friction accurately. Experimentally his work indicated that the temperature effect was small, temperature fluctuations of ±5°c had no measureable effect. He also examined the directional sensitivity of the hot-film and found that when the long edge of the hot-film is at right angles to the flow the directional sensitivity was the least and when parallel to the flow the sensitivity was the greatest. When the probe was aligned within ±15° of the normal there was no measureable difference in indicated skin friction. His experiments also showed that the requirement that the thermal boundary layer be contained within the viscous sublayer was not very stringent, and that some latitude was permitted in this regard. This work only considered the mean skin friction.

Liepman and Skinner [1954] used a slightly different heated element. In their work they used a fine hot-wire element embedded into a surface. Theoretically they used dimensional arguments and similarity to obtain the heat transfer relationship given previously. They felt that due to heat leakage the element would have to be calibrated, rendering a more complex analysis unjustified. Additionally, they showed that the instrument could be used in favourable pressure gradients and compressible flows without adverse influence, except near separation. They felt their experiments showed that a single calibration could be used extending from laminar to turbulent flow. Liepman and Skinner were the first to place limits on the size of the element though the relationship $Pr/Cf > Nu >> 1$. In words, the first condition requires that the viscous sublayer must be larger than the size of the thermal boundary layer and the second condition requires that the length be greater than the minimum length for a boundary layer approximation. As before, this work only considered the mean or steady heat transfer.

Bradshaw and Gregory [1961] examined the laminar to turbulent extention of the calibration of Stanton tubes, surface pitot tubes and hot-wires mounted just above a surface. The investigation, in part, was made to determine whether the laminar and turbulent flow calibrations were the same. They argued that if an instrument with a linear response to velocity fluctuations was introduced into the linear part of the profile, it could be calibrated in laminar flow and used accurately in turbulent flow. For the instruments they examined it was found that the laminar calibration was not the same as the turbulent calibration. This work has been used to repudiate Liepman and Skinner's assertion that the laminar calibration of a hot-film can be used in turbulent flows.

Bellhouse and Schultz [1966] were the first to work with thin-film heated elements. Their probe was an approximately 1 μm thick platinum-silver alloy film baked onto a Pyrex glass substrate. It was claimed that these films have a time constant of about 0.04 μsec and could be used for dynamic measurements. They produced two types of hot-film elements, but only the removable elements will be considered here. The removable probe could be withdrawn and replaced without great attention to relative surface displacement, and the readings were repeatable to a high order of accuracy. For about a hundred separate runs, they found that the 1/3 power of the shear stress was a linear function of the power dissipated. This is in agreement with all of the previous findings. For the thin-films, Bellhouse and Schultz found that even small changes in ambient temperature, as small as 0.1°C, had a significant influence on the measurements. This is contradictory with Ludwieg's findings on this point. They also found that the calibration in laminar flow and the calibration in a turbulent flow did not match. The response to shear stress was similar in both cases but the constants were different. This is in agreement with the conclusion of Bradshaw and Gregory that laminar calibrations cannot be extended to turbulent flows, at least near the solid surface. This is in disagreement with the findings of Liepman and Skinner. An interesting side point was that they had to use batteries to achieve acceptable noise levels when recording the fluctuating shear stresses from laminar separation.

Dynamic measurements were made of the shear stress fluctuations in a turbulent boundary layer. Using a dynamic calibration (below 100 (Hz) Bellhouse and Schultz claimed to be able to measure the shear stress spectrum. The dynamic calibration was performed by mounting the hot-film on a oscillating plate and comparing the readings with the skin friction calculated by Lighthill's [1954] formula. They found that the response was linear in the range that they measured. They did not state whether batteries were used to power the film for these measurements.

Bellhouse and Schultz derived the relationship for the heat transfer from a heated element through the thermal energy integral. This method is given below. The thermal energy integral is given by

$$d\{\int_0^\infty U(T-T_1)dy]\}/dx = -Q_w(x)/\rho Cp \qquad \ldots \text{B.1}$$

for steady two-dimensional flow with insignificant viscous heat production and Reynolds transport, where T_1 is the ambient fluid temperature. It is quite adequate to assume that the temperature distribution is given by

$$(T-T_1)/(T_w-T_1) = f(\eta) = 1+\lambda\eta+b\eta^2+c\eta^3+d\eta^4 \quad \ldots \text{B.2.a}$$

where $\eta=y/\delta_T$ and δ_T is the thermal boundary layer thickness. On applying the boundary conditions ($T=T_1$, $\partial T/\partial y=0$ at $\eta=1$ and $T=T_w$, $\partial T/\partial y=Q_w/k$, $\partial^2T/\partial y^2=0$ at $\eta=0$) one obtains

$$f(\eta) = (1-4\eta^3+3\eta^4)+\lambda(\eta-3\eta^3+2\eta^4) \qquad \ldots \text{B.2.b}$$

where $\lambda=Q_w\delta_T/k(T_w-T_1)$, the temperature profile shape parameter. In the viscous sublayer, we may assume that the Reynolds stresses are insignificant. Hence

$$dP/dy = \mu\partial^2U/\partial y^2 \qquad \ldots \text{B.3.a}$$

for two-dimensional flows. Integrating this equation twice and applying the boundary conditions that $U=0$ and $\partial U/\partial y=\tau/\mu$ at $y=0$ yeilds

$$U = \tau y/\mu + (y^2/2\mu)dP/dx. \qquad \ldots \text{ B.3.b}$$

substituting B.2 and B.3 into B.1 and carrying out the integration gives

$$d\{ak^2\tau(T_w-T_1)^3/Q_w^2\}/dx - d\{b(dP/dx)k^3(T_w-T_1)^4/2Q_w^3\}/dx$$
$$= -\mu^2 Q_w(x)/\rho k\mathrm{Pr}. \qquad \ldots\text{B.4.a}$$

where $a=\lambda^2\int_0^1\eta f(\eta)d\eta$ and $b=-\lambda^3\int_0^1\eta^2 f(\eta)d\eta$. According to Curle [1962] a and b may be treated as constants. Assuming that the temperature of the surface is equal to the ambient temperature except at the hot-film where it is a constant T_w and that Q_w is zero except for the hot-film where it is uniform, B.4.a can be integrated to give the net heat transfer, or power dissipation per unit depth. This results in

$$ak^2\tau(\Delta T_0/Q_w)^3 - (bk^3/2)(dP/dx)(\Delta T_0/Q_w)^4 = -\upsilon^2 L/\rho k\mathrm{Pr} \ldots \text{ B.4.b}$$

where $\Delta T_0=T_w-T_1$. Assuming that the pressure gradient term is sufficiently small gives

$$ak^2\tau(\Delta T_0/Q_w)^3 = -\mu^2 L/\rho k\mathrm{Pr} \qquad \ldots \text{ B.5.a}$$

or

$$\tau = -\mu^2 L(Q_w/\Delta T_0)^3/a\rho k^3\mathrm{Pr}. \qquad \ldots \text{ B.5.b}$$

Bellhouse and Schultz also gave the following expression for the temperature dependence of the hot-film calibration

$$I^2R/\Delta T_0 = \tau^{1/3}(A+B\Delta T_0)+(C+D\Delta T_0) \qquad \ldots \text{ B.6}$$

where A, B, C and D are constants. This was an experimental finding which they supported by a questionable theory. Experimentally they found that for small ΔT_0 variations the constant temperature calibration could be used provided that $I^2R/\Delta T_0$ was corrected. For ΔT_0 approximately 100 °C, a 1 °C fluctuation in ambient temperature would yeild a 1% error in the power dissipation term and roughly a 3% error in the indicated shear stress. This indicates the importance of stable ambient temperatures.

Brown [1967] used von Mises' transformation of the thermal boundary layer equation and an assumed quadratic velocity distribution to solve the heat transfer relationship in laminar flows. This was done to avoid the assumption that the temperature distribution was independent of the pressure gradient that is used in the integral analysis. He obtained asymptotic analytical solutions for small dP/dx and small τ. On comparison to the Bellhouse and Schultz relation B.4, he obtained

$$\tau+(15L/18Nu)(dP/dx)=\mu^2Nu^3/(0.8073)^3\rho L^2Pr \qquad \text{...B.7}$$

which is functionally similar. He concluded that with constants as in B.7 the integral solution

> 'is an adequate representation of the relationship between heat transfer skin friction and pressure gradient over the whole range of pressure gradients from a stagnation point to separation'.

Rather than use the measured length of the hot-film, Brown suggested that an effective length be calculated from the slope of a zero pressure gradient calibration. This avoids the top-hat temperature distribution assumption. He found that for his probe, in a laminar flow, the effective length was twice the measured length. In a turbulent flow it was three times the measured length.

Experimentally, Brown found that the pressure gradient could be ignored except for very near a stagnation point. This distance could be reduced by using a shorter gauge length.

For turbulent flows he used a Gaussian temperature distribution along the wall and a linear velocity distribution. He showed that by Laplace transforming the thermal boundary layer equation with respect to x an Airy equation could be obtained. From solutions of this equation Brown found that all of the heat is supplied to the air from the surface over $-\infty <(x-x_0)/L\leq 1.15$. Less that 10% of the heat supplied to the air crosses at the plane $(\rho Pr\tau L^2/\mu^2)^{1/3}(y/L)=3$. Thus if the linear part of the velocity distribution extends past this point the heat transfer should be unaffected by turbulence. Using $u_* h/\nu=12$ as the outer edge of the linear layer, Brown found that

$$u_* L/\nu < 64Pr \qquad \ldots \text{ B.8}$$

was the design criteria to have laminar calibrations equal the turbulent calibrations. He confirmed this with his experiments.

Pope [1972] carried out measurements on thin-film heated elements. His measurements of the temperature distribution along the surface about the film clearly showed the top-hat distribution to be incorrect. But, using the measured temperature distribution for a theoretical 'calibration' he found that the temperature distribution had little effect on the steady flow calibration. He felt that the difference between the calibrations in laminar and turbulent flow was the result of the different nature of the two boundary layers. Using linear models for the variation of dynamic viscosity μ and the thermal conductivity k with distance from the wall, he deduced that the effective viscosity had a slope given by $0.041u_*/\nu$. From this he developed the design critera

$$u_*L/\nu = 32Pr \qquad \ldots \text{B.9}$$

which was more stringent than Brown's.

Later, Geremia [1972] presented some experimental results of calibration measurements. DISA removeable probes were used, but the exact model number was not given. There were three primary objectives of this report; first could the 1/3 power law be used to calibrate these films, second would the calibration hold when the probes were used for long periods of time or were moved and finally could a calibration obtained in a pipe be used in a flat plate flow. Using 6 different probes he found that the probes were subject to the 1/3 power law. He also found that the calibration changed by less than 5% when the probes were used for extended periods of time or had been removed and later replaced. By comparing calibrations, Geremia concluded that the calibration in a pipe and the calibration on a flat plate were the same. One point that he raised was that the misalignment of the probe should be an order of magnitude less than the sublayer thickness. This misalignment is the result of the surface of the probe protruding above the surface of the wall. It is possible that either the sublayer structure or the response of the probe would be altered.

Reichert and Azad [1977] examined the calibration characteristics of removeable hot-film probes in developing turbulent pipe flow with varing temperature. They found that calibration equations such as B.6 fit the measured data well and that higher order terms were unnessesary. The results showed that the film probes could be used to measure a wide range of stresses, and that compensation for changes in temperature were very important. They found that the calibration changed when the probes were moved from one location to another. This was attributed to changes in the heat conduction from the probe body to the wall as a large proportion of the heat transfer is to the substrate.

Miller [1980] examined the position sensitivity of the hot-film elements to relative displacement to the surface. He found that small positional errors resulted in very large errors in the indicated shear stress. This work was of a qualitative nature and no tolerances were indicated for minimizing this error.

Up to this point, the work reported on has not considered, seriously, the measurement of the dynamic shear stresses. This is a point that has not been studied thoroughly as of now. The work that has been reported will be discussed presently.

Bellhouse and Schultz [1968] derived a theoretical frequency response for hot-film shear stress probes based on Lighthill's [1954] integral analysis of fluctuating wall shear stress and heat transfer. Their work was a simplification of the integral analysis. They restricted the response to above 200 Hz to eliminate substrate feedback. No reason was given for the 200 Hz limit, nor was any indication of its variation with fluid or substrate properties. For $\beta=(\omega\bar{\delta}_T^2/\kappa)^{1/2}$, they gave

$$\tau_m/\tau = [\{(L_1+L)^{2/3}-L_1^{2/3}\}/L_{eff}^{2/3}] / [1+2\beta^2 i/45] \quad \text{for } \beta \sim 1 \qquad \ldots \text{B.10}$$

$$\tau_m/\tau = 25.09[(L_{eff}/L_1)^{1/3}-(L_{eff}/L_1+L)^{1/3}] / (\sqrt{i}\beta)^3 \quad \text{for } \beta >> 4 \qquad \ldots \text{B.11}$$

where L_{eff} is the effective length of the film, L is the actual length and L_1 is the heated length upstream from the film. Equation B.10 was shown to fit the experimental data quite well if the bracketed term containing the lengths was 0.243.

Both Sandborn [1979] and Ramaprain and Tu [1983] indicate that the non-linear response cause time averaged measurements to be incorrect. Because of this they indicate special calibration procedures that correct for skewing due to time dependent quantities.

In an interesting article on hot-film anemometry Brison, Charnay and Comte-Bellot [1979] present the results of a finite element study of conical and wedge-shaped probes. They found that the substrate and the hot-film had a strong thermal interaction that affected the frequency response. Unfortunately this work was not done for flush mounted wall shear stress gauges, as this would have been of considerable interest.

In summary, we can see that hot-film shear stress probes have had a limited amount of study. There are several points of contention that are outstanding however. One issue that deserves more attention is the dynamic response of the gauges. Unfortunately too much faith is still required in using these instruments.

Appendix C

Data Acquisition

As this is the first work that used the digital data processing facilities at the University of Manitoba turbulence laboratory it is proper that some description of the data acquisition system be given. In the following discussion words printed in upper case refer to either a DEC (Digital Equipment Corporation) product, a programming language or an acronym. These terms will be defined if their meaning is unclear.

C.1 HARDWARE

Data acquisition is done using a LPA11-K (Laboratory Peripheral Accelerator, a DEC product). This device is mounted directly onto the UNIBUS (DEC data bus for PDP-11 computers) of the PDP-11/34a and provides an interface between the laboratory devices and the computer. Because of this the computer does not 'see' any of the laboratory devices and hence any effort to directly reference them would be unsuccessful. The data pathways and devices are shown schematically in figure C.1. This system allows very high data transfer rates with a minimum amount of servicing from the host processor.

The following is a brief description of how the LPA11-K operates, a thorough description is given by DEC [1978] which also contains information on writing the driver programs for it. The LPA11-K is comprised of four basic units, a master microprocessor, a slave microprocessor, an interprocessor buffer (IPB) and an I/O bus. The master microprocessor is connected directly to the UNIBUS and has access to the main memory for direct memory access (DMA) data transfers. It controls all of the transfers between the host processor and the LPA-11K which includes the control information and the inflowing and outflowing data. The slave microprocessor directly controls the laboratory devices. These two microprocessors are wired together through the IPB, which is a memory used to buffer the data between the master and slave microprocessors. Buffering is done so no data will be lost. Both the slave microprocessor and the IPB are connected to the I/O bus. The laboratory devices are connected to this bus.

Because of this structure, several advantages are obtained. The host computer only has to load the microcode for the microprocessors, maintain a request data table, provide a start and stop request and transfer relatively large buffers of data in the memory. When the laboratory devices are on the UNIBUS the host processor must continuously monitor the performance of the devices and load each sample into memory. For large data acquisition rates the rate can be limited because of hardware/software limitations of the host processor as well as preventing any concurrent data processing.

The laboratory devices on the I/O bus were a KW11-K (DEC dual programmable clock) and an AD11-K (DEC A/D converter). The clock is thoroughly described in DEC [1976] and the A/D converter in DEC [1977]. The KW-11K is comprised of two clocks, clock A a 16 bit programmable clock and clock B an 8 bit programmable clock. Both clocks can be set for one of five internal crystal-controlled rates (1MHz, 100kHz, 10 kHz, 1kHz and 100 Hz) or an external input Schmitt trigger. Clock A can also be set to operate from the clock B output. In operation a preset value (a negative integer) is loaded into a buffer and this buffer is incremented each time the internal clock pulses. Every time this buffer becomes zero an external timing pulse is made and the initial preset value is loaded back into the buffer and the cycle repeats. In this way a nearly continuously variable output clock rate may be obtained. This clock pulse can then be used to initiate an A/D conversion, a D/A conversion or a digital data transfer. The AD-11K is a single 12 bit A/D converter with a 16 channel multiplexer. The input signal must be within ±5 V. Each increment is 2.44 mV with -4.991 V being given by 0 digital output and 0 V corresponding to 2048(decimal). The multiplexer could be switched (internally) to perform differential measurements for eight channels. It was operated for sixteen single ended inputs or eight pseudo-differential inputs. For a single channel (no multiplexer switching) 75 ksample/sec could be obtained. This went down to an aggregate rate of 40 ksamples/sec for multichannel operations. Thus for two channels the maximum sampling rate would be 20 ksamples/sec on each channel. These samples would be made sequentially in time. With a second AD-11K the effective sampling rates could be doubled and parallel sampling could be done.

DEC [1981a] provides a very good overview of their peripherals. It is strongly recommended that anyone interested in these devices examine this reference.

The analog input signals were preconditioned before A/D conversion. A high quality amplifier with variable gain and offset and adjustable 18 db/oct low-pass filter was set to half the sampling rate. The variable gain and offset were used to force the output signal as close as possible to the ±5 V limits for the A/D converter. This signal was monitored on an oscilloscope.

C.2 SOFTWARE

To operate the LPA-11K, microcode must be loaded into the microprocessors. This is done either by the operating system or by direct program control. There are three versions of microcode that can be loaded, dedicated A/D, dedicated

D/A and multirequest. Because the facility only has an A/D capability and it is anticipated that only a single request will be made at any given time, dedicated A/D mode was selected at system generation and this is loaded when the system is booted in. The operating system (RSX-11M) contains MACRO and FORTRAN callable subroutines that can be used to make the request. The detailed description of these subroutines is given in DEC [1983] and additional information on the FORTRAN calls is in DEC [1979]. These subroutines are generally in the system library so no special request must be made when building the tasks that use them. Because of the functional structure of the LPA-11K it is not worth the effort to write one's own subroutines as only an improvement in the speed of the request and not the speed of operation will be made.

A FORTRAN program was written to collect data. It asked questions, through the terminal, that allowed most of the features of the devices to be selected and controlled at execution. Digital has provided some very good software in this regard, it was a simple matter to read their well written manuals and program the devices. Unfortunately the data transfer facilities, from FORTRAN, were too slow to keep up with the A/D converter. Using FORTRAN write statements only a 1 k sampling rate could be sustained. Because of this a set of MACRO subroutines, FORTRAN callable, were written. Details of using the system I/O operations are given in DEC

[1981b], where the information on block data transfers is important. These subroutines executed block data transfers of 5120 samples (5 k) each time. This corresponded to 1 sector on a RL01 disk. Dumping a sector at a time minimized the access time for each transfer. These transfers were buffered in that two 5120 word blocks of data were used to transfer the data. One block was being transfered to the disk while the other was being written to from the program. This also enhanced the data transfer rate. Because of the large amount of memory that was used, it was felt that the data collection program should only collect the data and write it to a disk. It would not do any signal processing.

The RL01 disks also had to be specially initialized. For the maximum capacity of 2.3 Mwords the index file had to be located at either the beginning of the end of the disk. For the index file at the beginning of the disk sustained (to capacity) sampling rates of 71,000 samples/sec have worked and for the index file at the end for the disk only sampling rates below 50,000 samples/sec can be sustained.

C.3 DATA ACQUISITION PROGRAMS

The following programs are the code used for the data acquisition. The MACRO subroutines are used to transfer the data in and out of memory with a disk.

C.3.1 The FORTRAN Data Collection Program

```
C-----------------------------------------------------------------------------
C
C            This program is used to collect data from the LPA11-K
C       laboratory peripheral accelerator.  The program is set up to
C       collect the data for very high frequency sampling rates for
C       large time periods.  It has been tested and found to work well
C       with sampling rates up to 71,000 samples/sec collected continuously
C       for 2300 buffers (1024 words each) with no loss of data.
C
C            The program requires that a disk mounted in DL01: has been
C       initialized to have its directory at either the beginning or the end
C       of the disk.  The default on the initialization of the directory at
C       the middle of the disk can sustain no more than a 1,000 sample/sec
C       data acquisition rate without failure.
C
C            The program is interactive and asks the issuing terminal a series
C       of questions.  In order to answer the questions the user must be
C       familiar with the LPA11-K terminology.
C
C-----------------------------------------------------------------------------
C
C       data declaration region
C
C-----------------------------------------------------------------------------
C
        LOGICAL L
C
C       L         - a logical used in answering some of the questions
C
        INTEGER IRATE,IPRSET,IND,IBUF(40),BUF(1024,4)
C
C       IRATE     - the integer code for the clock rate
C       IPRSET    - the negative value of the number of clock pulses between
C                   samples
C       IND       - an error indicator
C       IBUF      - a 40 word buffer used to initialize the LPA11-K
C       BUF       - 4 buffers (1024 words) for data storage
C
        INTEGER ICHN,INC,NCHN,I,J,IBUFNO,COUNT
C
C       ICHN      - the first channel number
C       INC       - the channel increment
C       NCHN      - the number of channels to sample
C       I,J       - dummy indexes
C       IBUFNO    - the next buffer that the program can write to disk
C       COUNT     - the number of buffers that have been written
C
        INTEGER ITAPE,ITCHN(16),IBAND,ISIZE,MSIZE
C
C       ITAPE     - the tape number (used for data identification)
C       ITCHN     - a 16 word buffer used to store the tape channel
C                   numbers (used for data identification)
C       IBAND     - the tape band number (used for data identification)
C       ISIZE     - the number of buffers (1024 words) of data to collect
C       MSIZE     - the maximum number of buffers that can be collected
C
        REAL RATE,ARATE,MRATE,DWELL,ADWELL
C
C       RATE      - the desired sampling rate
C       ARATE     - the actual sampling rate
C       MRATE     - the maximum sampling rate possible
C       DWELL     - the desired intersample time period
C       ADWELL    - the actual intersample time period
C
        REAL DCPART(16),ACPART(16)
```

```
C
C       DCPART   - a 16 (X2) word array used to store the DC part of the
C                  input signal (m/sec,etc.)
C       ACPART   - a 16 (X2) word array used to store the AC part of the
C                  input signal (m/sec,etc.)
C
C-----------------------------------------------------------------------
C
C       subroutine declaration region
C
        REAL XRATE,FASOPN,FASDMP,FASCLS

C
C       XRATE    - the subroutine that is used to calculate IRATE and
C                  IPRSET
C       FASOPN   - the subroutine that open the data file
C       FASDMP   - the subroutine that writes the data to the data file
C       FASCLS   - the subroutine that closes the data file
C
C-----------------------------------------------------------------------
C
C       this is the start of the code
C
C-----------------------------------------------------------------------
C
C       set the maximum sampling rate to 7.1E4
C
        MRATE=7.1E4
C
C       set the maximum number of buffers to collect to 2350
C
        MSIZE=2350
C
C       open the data file
C
        CALL FASOPN
C
C-----------------------------------------------------------------------
C
C       this block of code initializes the LPA11-K
C
C-----------------------------------------------------------------------
C
C       set the initialization buffer IBUF
C
        CALL SETIBF(IBUF,IND,,BUF(1,1),BUF(1,2),BUF(1,3),BUF(1,4))
C
C       if there is an error type a message and quit
C
        IF(IND.NE.0) GO TO 10
        TYPE 5000
5000    FORMAT(' SETIBF error')
        GO TO 230
10      CONTINUE
C
C       see if differential mode is required
C
        TYPE 5010
5010    FORMAT(' Do you want differential mode (T/F) ? ',$)
        ACCEPT 5020,L
5020    FORMAT(L1)
        IF(L) GO TO 20
C
C       if differential mode is not required set INC to 1
C
        INC=1
```

```
        GO TO 30
20      CONTINUE
C
C       if differential mode is required set INC to 2
C
        INC=2
C
C       remind the user that for differential mode ICHN must be even
C
        TYPE 5030
5030    FORMAT(' *** NOTE: differential mode requires the '
     *  ,'starting channel to be even ***')
30      CONTINUE
40      CONTINUE
C
C       ask for the starting channel
C
        TYPE 5040
5040    FORMAT(' What is the starting channel ? ',$)
        ACCEPT 5050,ICHN
5050    FORMAT(I3)
C
C       test to see if differential mode was required
C
        IF(.NOT.L) GO TO 50
C
C       test ICHN to see if it is even
C
        COUNT=ICHN/2
        COUNT=2*COUNT
        IF(COUNT.EQ.ICHN) GO TO 50
C
C       if ICHN is not even type a message and try again
C
        TYPE 5060
5060    FORMAT(' for differential mode the starting channel'
     *  ,' must be even, try again ')
        GO TO 40
50      CONTINUE
C
C       see how many channels are to be sampled
C
        TYPE 5070
5070    FORMAT(' How many channels are to be sampled ? ',$)
        ACCEPT 5050,NCHN
C
C       set the A/D converter
C
        CALL SETADC(IBUF,,ICHN,NCHN,INC,IND)
C
C       make sure that SETADC worked properly
C
        IF(IND.NE.0) GO TO 60
C
C       if it did not type a message and quit
C
        TYPE 5080
5080    FORMAT(' SETADC did not work.')
        GO TO 230
60      CONTINUE
C
C-----------------------------------------------------------------------
C
C        This block of code asks the user for a sampling rate
C        for the LPA11-K as well as which clock to use. The
C        actual sampling rate is reported back to the user
```

```
C         then the clock is set and a status report is made.
C
C-----------------------------------------------------------------------
C
C       ask which clock to use
C
        TYPE 5090
5090    FORMAT(' Do you want clock A (T) or clock B (F) ? ',$)
        ACCEPT 5020,L
70      CONTINUE
C
C       ask for the sampling rate that is to be used
C
        TYPE 5100
5100    FORMAT(' What sampling rate is required (samples/sec) ? ',$)
        ACCEPT 5110,RATE
5110    FORMAT(E15.3)
C
C       check the sampling rate to make sure it is a valid rate
C
        IF (RATE.GT.0.0) GO TO 80
C
C       if the rate is negative type a message and try again
C
        TYPE 5120
5120    FORMAT(' Negative sampling rate , try again.')
        GO TO 70
80      CONTINUE
        IF(RATE.LE.MRATE) GO TO 90
C
C       if the rate is to high type a message and set it to the
C       maximum rate
C
C       NOTE: The maximum sampling rate for a single A/D converter is
C       71,000 samples/sec, to change the maximum rate change MRATE.
C
        RATE=MRATE
        TYPE 5130,MRATE
5130    FORMAT(' Sampling rate to large, it was set to ',E9.2)
90      CONTINUE
C
C       ask how many buffers are to be filled and set it to the multiple
C       of 5 just greater than required
C
        TYPE 5140
5140    FORMAT(' How many buffers are to be filled ? ',$)
        ACCEPT 5150,ISIZE
5150    FORMAT(I5)
        ISIZE=ISIZE/5
        ISIZE=ISIZE+1
        ISIZE=5*ISIZE
C
C       test to see if ISIZE is permissable
C
        IF(ISIZE.LE.MSIZE) GO TO 100
C
C       if it is to large set it to MSIZE and type a message
C
        TYPE 5160,MSIZE
5160    FORMAT(' The number of buffers was to large it was set to',2X,I4)
100     CONTINUE
C
C       convert the sampling rate to the dwell
C
        DWELL=1.0/RATE
C
C       if clock A was chosen skip this step
C
        IF (L) GO TO 110
```

```
C
C       call XRATE to set the actual intersample time period, IRATE and
C       IPRSET for clock B
C
        ADWELL=XRATE(DWELL,IRATE,IPRSET,1)
C
C       calculate the actual sampling rate
C
        ARATE=1.0/ADWELL
C
C       start clock B
C
        CALL CLOCKB(IRATE,IPRSET,2,IND,7)
C
C       if clock B did not start properly type a message and quit
C
        IF(IND.NE.0) GO TO 120
        TYPE 5170
5170    FORMAT(' Clock B did not start properly.')
        GO TO 230
110     CONTINUE
C
C       call XRATE to set the actual intersample time period, IRATE
C       IPRSET for clock A
C
        ADWELL=XRATE(DWELL,IRATE,IPRSET,0)
C
C       calculate the actual sampling rate
C
        ARATE=1.0/ADWELL
C
C       start clock A
C
        CALL CLOCKA(IRATE,IPRSET,IND,7)
        IF(IND.NE.0) GO TO 120
C
C       if clock A did not start properly type a message and quit
C
        TYPE 5180
5180    FORMAT(' Clock A did not start properly.')
        GO TO 230
120     CONTINUE
C
C       type out the sampling information
C
        TYPE 5190
5190    FORMAT('0',80('*')/' ')
        TYPE 5200
5200    FORMAT(' ','The clock started sucessfully.')
        TYPE 5210,RATE,ARATE,DWELL,ADWELL
5210    FORMAT('0',T10,'Desired rate',2X,E11.3,2X,'(sample/sec)'
     *  /' ',T10,'Actual rate',3X,E11.3,2X,'(samples/sec)'/' ',
     *  T10,'Desired dwell',2X,E11.3,2X,'(sec)'/' ',T10,
     * 'Actual dwell',3X,F11.3,2X,'(sec)')
        TYPE 5230,IRATE,IPRSET
5230    FORMAT(' ',T10,'IRATE',9X,I2/' ',T10,'IPRSET',8X,I5)
        TYPE 5190
C
C-----------------------------------------------------------------------
C
C       this section sets the data identification information
C
C-----------------------------------------------------------------------
C
        TYPE 5240
5240    FORMAT(' Which tape is the data from? ',$)
        ACCEPT 5050,ITAPE
        DO 130 I=1,NCHN
```

```
        TYPE 5250,I
5250    FORMAT(' For input channel ',I2,' the tape channel is ? ',$)
        ACCEPT 5050,ITCHN(I)
130     CONTINUE
        TYPE 5260
5260    FORMAT(' Which tape band is the data from ? ',$)
        ACCEPT 5050,IBAND
        DO 140 I=1,NCHN
        TYPE 5270,I
5270    FORMAT(' For input channel ',I2,' the DC part was ? ',$)
        ACCEPT 5280,DCPART(I)
5280    FORMAT(F8.4)
        TYPE 5290,I
5290    FORMAT(' For input channel ',I2,' the AC part was ? ',$)
        ACCEPT 5280,ACPART(I)
140     CONTINUE
C
C----------------------------------------------------------------------
C
C       this the preliminary information has been collected
C
C       now the program can start collecting data when desired
C
C----------------------------------------------------------------------
C
C       halt the program for a start indicator
C
        TYPE 5300
5300    FORMAT(' To start the data collection hit a carrage'
     *  ,' return. ',$)
        ACCEPT 5020,I
C
C----------------------------------------------------------------------
C
C       release the data buffers and start the sweep
C
C----------------------------------------------------------------------
C
C       release the data buffers
C
        CALL RLSBUF(IBUF,IND,0,1,2,3)
        IF(IND.NE.0) GO TO 150
C
C       if RLSBUF did not work type a message and quit
C
        TYPE 5310
5310    FORMAT(' RLSBUF did not work.')
        GO TO 230
150     CONTINUE
C
C       start the sweep
C
        CALL ADSWP(IBUF,1024,0,0,,0,0,ICHN,NCHN,IND)
        IF(IND.EQ.1) GO TO 170
C
C       if the error indicator equals 1 everything worked
C
        IF(IND.LT.0) GO TO 160
C
C       if the error indicator was possitive a calling error was made
C       type a message and quit
C
        TYPE 5320
5320    FORMAT(' ADSWP calling error.')
        GO TO 230
160     CONTINUE
C
C       if the error indicator was negative a 100 error was made
C       type a message and quit
```

```
C
        TYPE 5330
5330    FORMAT(' 100 error.')
        TYPE 5340,IBUF(1),IBUF(2)
5340    FORMAT(' ','Fortran error code',2X,I4/' ',
     * 'LPA11-K error code',2X,O12)
170     CONTINUE
C
C       set the number of buffers written to disk equal to zero
C
        COUNT=0
180     CONTINUE
C
C       wait for for the next buffer to filled
C
        CALL IWTBUF(IBUF,0,IBUFNO)
C
C       if IBUFNO is negative type a message and quit
C
        IF(IBUFNO.LT.0) GO TO 200
C
C       write out the buffer
C
        CALL FASDMP(BUF(1,IBUFNO+1),32)
C
C       increament the counter
C
        COUNT=COUNT+1
C
C       release the buffer that has been written out to the disk
C
        CALL RLSBUF(IBUF,IND,IBUFNO)
        IF(IND.NE.0) GO TO 190
        TYPE 5230
        GO TO 230
190     CONTINUE
C
C       if the counter is less than equal to the number of buffers to be
C       collected than collect some more data
C
        IF(COUNT.LE.ISIZE) GO TO 180
C
C       if the counter is greater than the number of buffers to be collected
C       stop the sweep
C
        CALL STPSWP(IBUF,0,)
C
C       jump past the error handling code
C
        GO TO 210
200     CONTINUE
        TYPE 5350
5350    FORMAT(' Error in data collection.')
        TYPE 5340,IBUF(1),IBUF(2)
        GO TO 230
210     CONTINUE
C
C       on completion of the data collection send a message to the
C       issuing terminal
C
        TYPE 5360
5360    FORMAT(' Data collection complete.')
C
C       close the data file
C
        CALL FASCLS
C
C------------------------------------------------------------------------
```

```
C
C       this section opens an information file on the data storage disk
C       that is used to store the data collection information and the
C       data identification information.
C
C------------------------------------------------------------------------
C
C       open the information file
C
        OPEN(UNIT=3,TYPE='NEW',NAME='DL1:INFO.DAT')
C
C       write the number of channels sampled,the tape number
C       the band number
C
        WRITE(3,*) NCHN,ITAPE,IBAND
C
C       write the tape channel numbers
C
        WRITE(3,*) (ITCHN(I),I=1,NCHN)
C
C       write the actual intersample time period
C
        WRITE(3,*) ADWELL
C
C       write the DC and AC parts of the input signals
C
        DO 220 I=1,NCHN
        WRITE(3,*) DCPART(I),ACPART(I)
220     CONTINUE
C
C       write the number of 1024 word buffers that were collected and
C       writen into the data file
C
        WRITE(3,*) ISIZE
C
C       close the information file
C
        CLOSE(UNIT=3)
230     CONTINUE
C
C       this is the end of the program
C
        STOP
        END
```

C.3.2 The Fast Output Subroutines

These subroutines perform three functions; open a file on unit 8, write out the entire sectors on a RL01 disk drive, and close the file. The output transfers are buffered with two 1 sector (5210 word) blocks so one block is available to accept data while the write operation is being performed. Thus no data will be lost.

C.3.2.1 Subroutine FASOPN

```
        .TITLE  FASOPN
        .IDENT /FV01/
;
;****************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT OPENS A LARGE CONTIGUOUS FILE ON LOGICAL UNIT 8.
;       THE INITIAL FILE SIZE IS 9600 BLOCKS, AND THE FILE
;       IS EXTENDED BY 120. BLOCKS IN EACH EXTENTION.  THIS
;       SUBROUTINE (ALONG WITH ITS SIBLINGS) CAN PROVIDE
;       DATA TRANSFER RATES OF 71 K W/MIN WITH THE LPA11-K
;       AND A RL01 CONTINUOUS WITH NO LOSS OF DATA.
;
;       NOTE:   FOR OPTIMUM PERFORMANCE THE DISK SHOULD BE
;               INITIALIZED TO HAVE ITS DIRECTORY AT EITHER
;               THE BEGINING OR THE END OF THE DISK.
;
;               DO NOT USE EVENT FLAG 22(DECIMAL) FOR ANY
;               OTHER APPLICATION, IT IS USED TO COORDINATE
;               I/O.
;
;****************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;****************************************************************
;
        .GLOBL  FASOPN                  ; ADDRESS OF FASOPN SUBROUTINE
        .GLOBL  FDBOUT                  ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL  $SAVAL                  ; ADDRESS OF REGISTER SAVING SUBROUTINE
        .GLOBL  VBNADR                  ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
        .GLOBL  ENDBUF                  ; ADDRESS OF THE END OF THE INTERMEDIATE
                                        ;   BUFFER ADDRESS
        .GLOBL  BKDA                    ; ADDRESS OF THE BEGINING OF THE INTER-
                                        ;   MEDIATE BUFFER ADDRESS
        .GLOBL  DFNB                    ; ADDRESS OF THE DEFAULT NAME BLOCK
        .GLOBL  DSPT                    ; ADDRESS OF THE DATA SET DESCRIPTOR
        .GLOBL  IOST                    ; ADDRESS OF THE I/O STATUS BLOCK
        .GLOBL  PBLOC                   ; ADDRESS OF THE FIRST FREE WORD OF
                                        ; THE INTERMEDIATE BUFFER
;
;****************************************************************
;
;       MACRO CALL DEFINITIONS (ASSEMBLY-TIME)
;
;****************************************************************
;
        .MCALL  FDBDF$                  ; ALLOCATE FILE DESCRIPTOR BLOCK MACRO
        .MCALL  FDAT$A                  ; INITIALIZE FILE ATTRIBUTE SECTION MACRO
        .MCALL  FDRC$A                  ; INITIALIZE RECORD ACCESS SECTION MACRO
        .MCALL  FDBK$A                  ; INITIALIZE BLOCK ACCESS SECTION MACRO
        .MCALL  FDOP$A                  ; INITIALIZE FILE-OPEN SECTION MACRO
        .MCALL  NMBLK$                  ; DEFAULT FILE NAME BLOCK MACRO
;
;****************************************************************
;
;       MACRO CALL DEFINITIONS (RUN TIME)
;
;****************************************************************
;
        .MCALL  OPEN$W                  ; OPEN TO WRITE BLOCKS MACRO
        .MCALL  WRITE$                  ; WRITE BLOCKS MACRO
        .MCALL  FDBK$R                  ; INITIALIZE BLOCK ACCESS SECTION MACRO
;
;****************************************************************
```

```
;
;       START OF FASOPN CODE
;
;**********************************************************************
;
FASOPN: JSR PC,$SAVAL                   ; SAVE ALL OF THE REGISTERS
        MOV #ENDBUF,ENDBUF              ; STORE THE ADDRESS OF THE END OF THE
        SUB #2,ENDBUF                   ; INTERMEDIATE BUFFER IN ENDBUF
        MOV #BKDA,FBLOC                 ; STORE THE ADDRESS OF THE BEGINING OF
                                        ; INTERMEDIATE BUFFER IN FBLOC
        CLR VBNADR                      ; INITIALIZE THE VIRTUAL BLOCK NUMBER
        MOV #1,VBNADR+2                 ; TO 1(DECIMAL)
;
;       OPEN THE FILE ON LOGICAL UNIT 8. FOR BLOCK I/O
;
        OPEN$W #FDBOUT,#8.,#DSPT,#FD.RWM
;
;       THIS STEP IS REQUIRED TO INITIALIZE THE VIRTUAL BLOCK NUMBER
;       REGION OF THE FILE DESCRIPTOR BLOCK (FOR BLOCK I/O ONLY)
;
        FDBK$R #FDBOUT,#BKDA,#10240.,#VBNADR,#22.,#IOST
;
;       THE FOLLOWING WRITE, WRITES NO BLOCKS IT IS REQUIRED FOR EVENT
;       TIMING OF THE FASDMP SUBROUTINE
;
        WRITE$ #FDBOUT,#BKDA,#0,#VBNADR,#22.,#IOST
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
;
;**********************************************************************
;
;       THE FOLLOWING SECTION IS REQUIRED TO ESTABLISH THE FILE
;       DESCRIPTOR BLOCK
;
;**********************************************************************
;
;       ALLOCATE FILE DESCRIPTOR BLOCK STARTING AT FDBOUT
;
FDBOUT: FDBDF$
;
;       INITIALIZE FILE ATTRIBUTES SECTION OF THE FDB
;               -FIXED RECORDS
;               -RECORD IS TO BE PRECEDED BY <LF> AND FOLLOWED BY <CR>
;               -RECORD SIZE OF 10240. BYTES
;               -ALLOCATE 9600. CONTIGUOUS BLOCKS UPON OPEN
;               -EXTENT THE FILE BYE 120. BLOCKS PER EXTENTION
;
        FDAT$A R.FIX,FD.CR,10240.,9600.,120.
;
;       INITIALIZE RECORD ACCESS SECTION OF THE FDB FOR BLOCK I/O
;
        FDRC$A FD.RWM
;
;       INITIALIZE BLOCK ACCESS SECTION OF THE FDB
;               -BUFFER FOR BLOCK I/O STARTS AT ADDRESS BKDA
;               -BUFFER FOR BLOCK I/O IS 10240. BYTES LONG
;               -THIRD ARGUMENT IS A DUMMY ARGUMENT (FOR COMPATABILITY
;                       WITH FDBK$R )
;               -EVENT FLAG 22. IS TO BE USED FOR I/O COORDINATION
;               -I/O STATUS BLOCK IS AT ADDRESS IOST
;
        FDBK$A BKDA,10240.,,22.,IOST
;
;       INITIALIZE FILE-OPEN SECTION OF FDB
```

```
;               -LOGICAL UNIT NUMBER IS 8.
;               -DATA SET POINTER ADDRESS IS DSPT
;               -DEFAULT FILE NAME BLOCK ADDRESS IS DFNB
;               -A NEW FILE IS TO BE OPENED FOR WRITING
;
        FDOP$A 8.,DSPT,DFNB,FO.WRT
;
;       CREATE DATA SET FOR A FILE NAMED BLOCK.DAT
;
DSPT:   .WORD 0,0
        .WORD 0,0
        .WORD ONAMSZ,ONAM
ONAM:   .ASCII /BLOCK.DAT/
ONAMSZ=.-ONAM
        .EVEN
;
;       CREATE DEFAULT FILE NAME BLOCK
;               - NAME IS BLOCK
;               - FILE TYPE IS DAT
;               - VERSION 1
;               - DEVICE DL (RL01)
;               - UNIT 1
;
DFNB:   NMBLK$ BLOCK,DAT,1,DL,1
        .EVEN
;
;       ESTABLISH I/O STATUS BLOCK
;
IOST:   .WORD 0,0
        .END
```

C.3.2.2 Subroutine FASDMP

```
        .TITLE FASDMP
        .IDENT /FV01/
;
;************************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT COORDINATES THE WRITING OF DATA, AT HIGH SPEED, TO
;       A LARGE FILE. THIS SUBROUTINE (ALONG WITH ITS SIBLINGS)
;       CAN PROVIDE DATA TRANSFER RATES OF 71 K W/SEC WITH THE
;       LPA11-K AND A RL01 CONTINUOUS WITH NO LOSS OF DATA.
;
;       NOTE:   FOR OPTIMUM PERFORMANCE THE DISK SHOULD BE
;               INITIALIZED TO HAVE ITS DIRECTORY AT EITHER
;               THE BEGINING OR THE END OF THE DISK.
;
;               DO NOT USE EVENT FLAG 22(DECIMAL) FOR ANY
;               OTHER APPLICATION, IT IS USED TO COORDINATE
;               I/O.
;
;       CALLING CONVENTIONS:
;
;       FORTRAN         CALL FASDMP(BUF,N)
;
;       ARGUMENTS       BUF - THE INTEGER BUFFER TO BE TRANSFERED
;                       N   - THE NUMBER OF 32 WORD MULTIPLES OF
;                             THE DIMENSION OF BUF
;
```

```
;       NOTES:  THE SUBROUTINE FASDMP WRITES WHOLE SECTORS ON TO A
;               RL01 DISK (40 BLOCKS-10240.BYTES-5120.WORDS) USING
;               TWO INTERMEDIATE BUFFERS OF THIS SIZE. NO TRANSFER
;               IS ACTUALLY DONE UNLESS THE BUFFER IS FULL.  THUS
;               THE AMOUNT OF DATA TRANSFERED SHOULD BE IN MULTIPLES
;               OF 5120. WORDS OR THE TRANSFER WILL BE TRUNCATED TO
;               A MULTIPLE OF 5120. WORDS.
;
;*****************************************************************
;
;       GLOBLE SYMBOL DEFINITIONS
;
;*****************************************************************
;
        .GLOBL  FASDMP                  ; ADDRESS OF THE FASDMP SUBROUTINE
        .GLOBL  FDBOUT                  ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL  $SAVAL                  ; ADDRESS OF THE REGISTER SAVING SUBROUTINE
        .GLOBL  VBNADR                  ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
        .GLOBL  ENDBUF                  ; ADDRESS OF THE END OF THE INTERMEDIATE
                                        ;   BUFFER ADDRESS
        .GLOBL  BKDA                    ; ADDRESS OF THE BEGINING OF THE INTER-
                                        ;   MEDIATE BUFFER ADDRESS
        .GLOBL  PBLOC                   ; ADDRESS OF THE NEXT WORD OF THE INTER-
                                        ;   MEDIATE BUFFER TO BE FILLED
        .GLOBL  IOST                    ; ADDRESS OF THE I/O STATUS BLOCK
;
;*****************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;*****************************************************************
;
        .MCALL  WRITE$                  ; BLOCK I/O WRITE MACRO
        .MCALL  WAIT$                   ; WAIT FOR I/O TRANSFER MACRO
;
;*****************************************************************
;
;       START OF FASDMP CODE
;
;*****************************************************************
;
FASDMP: JSR     PC,$SAVAL               ; SAVE ALL OF THE REGISTERS
        MOV (R5)+,R0                    ; GET NUMBER OF ARGUMENTS (UNNESSESARY)
        MOV (R5)+,R0                    ; GET STARTING ADDRESS OF DATA BLOCK
        MOV @(R5)+,R1                   ; GET THE SIZE OF THE DATA BLOCK IN 32.
                                        ; WORD MULTIPLES
        MOV PBLOC,R2                    ; PUSH THE LOCATION OF THE FIRST FREE WORD
                                        ; OF THE INTERMEDIATE BUFFER INTO R2
PACK:   .REPT 32.
        MOV (R0)+,(R2)+                 ; MOVE THE DATA FROM CALLING BUFFER
        .ENDR                           ; TO THE INTERMEDIATE BUFFER
        SOB R1,PACK
        MOV R2,PBLOC                    ; STORE THE NEW ADDRESS OF THE FIRST FREE
                                        ; WORD IN PBLOC
        SUB #BKDA,R2                    ; GET NUMBER OF DATA POINTS
                                        ; STORED IN THE INTERMEDIATE
                                        ; BUFFER
        SUB #10240.,R2                  ; TEST TO SEE IF THE FIRST BUFFER IS FULL
        BLT EXIT                        ; IF IT IS NOT RETURN CONTROL TO THE CALLING
                                        ; PROGRAM
FIRST:  BGT SECOND                      ; IF THE ADDRESS OF THE FIRST FREE WORD IS IN
                                        ; THE SECOND BUFFER BRANCH TO SECOND
;
;       WAIT FOR THE LAST WRITE (THIS IS WHY A WRITE HAD TO BE ISSUED UPON
;       OPENING THE FILE). NOTE WHILE THE FIRST BUFFER IS BEING WRITEN ONTO
;       THE DISK THE SECOND BUFFER IS BEING FILLED.  SO IN MOST CASES THE
;       BUFFER HAS BEEN WRITEN ONTO THE DISK BEFOR THIS STEP IS PERFORMED
;               -FDB FDBOUT
;               -EVENT FLAG 22. USED IN I/O COORDINATION
;               -I/O STATUS BLOCK AT ADDRESS IOST
;
        WAIT$ #FDBOUT,#22.,#IOST
```

```
 ;
 ;       WRITE THE FIRST BUFFER
 ;               -BUFFER FOR BLOCK I/O STARTS AT ADDRESS BKDA
 ;               -BUFFER FOR BLOCK I/O IS 10240. BYTES LONG
 ;               -VIRTUAL BLOCK NUMBER AT ADDRESS VBNADR
 ;               -EVENT FLAG 22. IS USED FOR I/O COORDINATION
 ;               -I/O STATUS BLOCK IS AT ADDRESS IOST
 ;
         WRITE$  #FDBOUT,#BKDA,#10240.,,#VBNADR,#22.,,#IOST
         ADD #20.,VBNADR+2        ; INCREMENT THE VIRTUAL BLOCK
                                  ; NUMBER BY 20.
         BR EXIT           ; RETURN CONTROL TO THE CALLING PROGRAM
SECOND:  SUB #10240.,R2    ; TEST TO SEE IF THE SECOND BUFFER IS FULL
         BLT EXIT          ; IF IT IS NO RETURN TO THE CALLING PROGRAM
 ;
 ;       WAIT FOR THE LAST WRITE TO COMPLETE
 ;               -EVENT FLAG 22. IS TO BE USED FOR I/O COORDINATION
 ;               -I/O STATUS BLOCK AT IOST
 ;
         WAIT$ #FDBOUT,#22.,,#IOST
 ;
 ;       WRITE SECOND BUFFER
 ;               -BUFFER STARTS AT ADDRESS BKDA1
 ;               -BUFFER IS 10240. BYTES LONG
 ;               -VIRTUAL BLOCK NUMBER IS AT ADDRESS VBNADR
 ;               -EVENT FLAG 22. IS USED FOR I/O COORDINATION
 ;
         WRITE$ #FDBOUT,#BKDA1,#10240.,,#VBNADR,#22.,,#IOST
         MOV #BKDA,FBLOC          ; RESET THE FIRST FREE WORD OF THE BUFFER
                                  ; TO BKDA
         ADD #20.,VBNADR+2        ; INCREMENT THE VIRTUAL BLOCK NUMBER BY
                                  ; 20.

EXIT:    RTS     PC
BKDA:    .BLKB 10240.             ; RESERVE 10240. BYTES FOR THE FIRST
                                  ; BUFFER
BKDA1:   .BLKB 10240.             ; RESERVE 10240. BYTES FOR THE SECOND
                                  ; BUFFER IMMEDIATLY AFTER THE FIRST BUFFER
ENDBUF:  .WORD 0                  ; RESERVE A WORD FOR THE END OF BUFFER
                                  ; ADDRESS
FBLOC:   .WORD 0                  ; RESERVE A WORD TO STORE THE LOCATION
                                  ; OF THE FIRST FREE WORD IN THE INTER-
                                  ; MEDIATE BUFFER
VBNADR:  .BLKW 2                  ; RESERVE TWO WORDS FOR THE VIRTUAL
                                  ; BLOCK NUMBER
 ;
 ;       THIS IS THE END OF THE SUBROUTINE
 ;
         .END
```

C.3.2.3 Subroutine FASCLS

```
         .TITLE FASCLS
         .IDENT /FV01/
;
;***********************************************************************
;
;        THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;        IT CLOSE THE FILE OPENED BY FASOPN.
;
;        ARGUMENTS        NONE
;
;***********************************************************************
;
;        GLOBAL SYMBOL DEFINITIONS
;
;***********************************************************************
;
```

```
        .GLOBL FASCLS                   ; ADDRESS OF FASCLS SUROUTINE
        .GLOBL FDBOUT                   ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL                   ; ADDRESS OF THE REGISTER SAVING ROUTINE
;
;******************************************************************
;
;       MACRO CALL DEFINITIONS (RUN TIME)
;
;******************************************************************
;
        .MCALL CLOSE$                   ; CLOSE MACRO
;
;******************************************************************
;
;       START OF FASCLS CODE
;
;******************************************************************
;
FASCLS: JSR     PC,$SAVAL               ; SAVE ALL OF THE REGISTERS
        CLOSE$ #FDBOUT                  ; CLOSE THE FILE POINTED TO BY THE FILE
                                        ; DESCRIPTOR BLOCK AT ADDRESS FDBOUT
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
        .END
```

C.3.3 The Subroutines that read the output file

These subroutines are used to read the unstructured file created by the subroutines in the previous section. The standard file handlers can not read it. They perform the three operations of opening, reading and closing the file. Two versions of the subroutines are available. The SLWxxx subroutines handle up to a 5120 word buffer and the SMGxxx subroutines handle up to a 1024 word buffer.

C.3.3.1 Subroutine SLWOPN

```
        .TITLE SLWOPN
        .IDENT /FV01/
;
;******************************************************************
```

```
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT OPENS THE LARGE CONTIGUOUS FILE ON LOGICAL UNIT 8
;       THAT WAS CREATED BY FASOPN AND FILLED BY FASDMP.  THIS
;       SUBROUTINE (ALONG WITH ITS SIBLINGS ) IS USED TO READ
;       THE BLOCK STRUCTURED FILE THAT WAS CREATED FOR HIGH
;       SPEED DATA TRANSFER FROM A LPA11-K TO A RL01 OF LARGE
;       QUANTITIES OF CONTIUGOUS DATA.
;
;       NOTE:   A FILE MUST HAVE BEEN CREATED BY THE FASOPN,
;               FASDMP,FASCLS FAMILY OF SUBROUTINES IN ORDER
;               FOR THIS FAMILY OF SUBROUTINES TO WORK.
;
;               DO NOT USE EVENT FLAG 23(DECIMAL) FOR ANY OTHER
;               APPLICATION, IT IS USED TO COORDINATE I/O.
;
;**********************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;**********************************************************************
;
        .GLOBL SLWOPN           ; ADDRESS OF THE SLWOPN SUBROUTINE
        .GLOBL FDBIN            ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL           ; ADDRESS OF REGISTER SAVING SUBROUTINE
        .GLOBL VBN              ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
        .GLOBL END              ; ADDRESS OF THE END OF THE INTERMEDIATE
                                ;   BUFFER ADDRESS
        .GLOBL START            ; ADDRESS OF THE BEGINING OF THE INTER-
                                ;   MEDIATE BUFFER
        .GLOBL LOC              ; ADDRESS OF THE NEXT FREE LOCATION IN
                                ;   THE INTERMEDIATE BUFFER
        .GLOBL DFBN             ; ADDRESS OF THE DEFAULT NAME BLOCK
        .GLOBL DSTP             ; ADDRESS OF THE DATA SET DESCRIPTOR
        .GLOBL IOBK             ; ADDRESS OF THE I/O STATUS BLOCK
;
;************************************************************************
;
;       MACRO CALL DEFINITIONS (ASSEMBLY-TIME)
;
;************************************************************************
;
        .MCALL FDBDF$           ; ALLOCATE FILE DESCRIPTOR MACRO
        .MCALL FDAT$A           ; INITIALIZE FILE ATTRIBUTE MACRO
        .MCALL FDRC$A           ; INITIALIZE RECORD ACCESS SECTION MACRO
        .MCALL FDBK$A           ; INITIALIZE BLOCK ACCESS SECTION MACRO
        .MCALL FDOP$A           ; INITIALIZE FILE-OPEN SECTION MACRO
        .MCALL NMBLK$           ; DEFAULT FILE NAME MACRO
;
;************************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;************************************************************************
;
        .MCALL OPEN$R           ; OPEN TO READ BLOCKS MACRO
        .MCALL FDBK$R           ; INITIALIZE BLOCK ACCESS SECTION MACRO
;
;************************************************************************
;
;       START OF SLWOPN CODE

;
;************************************************************************
;
SLWOPN: JSR PC,$SAVAL           ; SAVE ALL OF THE REGISTERS
        MOV #END,END            ; STORE THE ADDRESS OF THE END OF THE
        SUB #2,END              ; INTERMEDIATE BUFFER IN END
        MOV #START,LOC          ; STORE THE ADDRESS OF THE BEGINING OF
                                ; THE INTERMEDIATE BUFFER IN LOC
        CLR VBN                 ; INITIALIZE THE VIRTUAL BLOCK NUMBER
        MOV #1,VBN+2            ; TO 1(DECIMAL)
```

```
;
;       OPEN THE FILE ON LOGICAL UNIT 8. FOR BLOCK I/O
;
        OPEN$R #FDBIN,#8.,#DSTP,#FD.RWM
;
;       THIS STEP IS REQUIRED TO INITIALIZE THE VIRTUAL BLOCK NUMBER
;       REGION OF THE FILE DESCRIPTOR BLOCK (FOR BLOCK I/O ONLY)
;
        FDBK$R #FDBIN,#START,#10240.,,#VBN,#23.
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
;
;*************************************************************************
;
;       THE FOLLOWING SECTION IS REQUIRED TO ESTABLISH THE FILE
;       DESCRIPTOR BLOCK
;
;*************************************************************************
;
;       ALLOCATE FILE DESCRIPTOR BLOCK STARTING AT FDBIN
;
FDBIN:  FDBDF$
;
;       INITIALIZE FILE ATTRIBUTES SECTION OF THE FDB
;               -FIXED RECORDS
;               -RECORD IS TO BE PRECEDED BY <LF> AND FOLLOWED BY <CR>
;               -RECORD SIZE OF 10240. BYTES
;               -INITIAL SIZE DEFAULTED TO FILE SIZE
;               -NO EXTENTIONS ON AN OPEN FOR READ
;
        FDAT$A R.FIX,FD.CR,10240.
;
;       INITIALIZE RECORD ACCESS SECTION OF THE FDB FOR BLOCK I/O
;
        FDRC$A FD.RWM
;
;       INITIALIZE BLOCK ACCESS SECTION OF THE FDB
;               -BUFFER FOR BLOCK I/O STARTS AT ADDRESS START
;               -BUFFER FOR BLOCK I/O IS 10240. BYTES LONG
;               -THIRD ARGUMENT IS A DUMMY ARGUMENT  ( FOR COMPATABILITY
;                       WITH FDBK$R )
;               -EVENT FLAG 23. IS TO BE USED FOR I/O COORDINATION
;               -I/O STATUS BLOCK IS AT ADDRESS IOBK
;
        FDBK$A START,10240.,,23.,IOBK
;
;       INITIALIZE FILE OPEN SECTION OF THE FDB
;               -LOGICAL UNIT NUMBER IS 8.
;               -DATA SET POINTER ADDRESS IS DSTP
;               -DEFAULT FILE NAME BLOCK ADDRESS IS DFBN
;               -AN OLD FILE IS TO BE OPENED FOR READING
;
        FDOP$A 8.,DSTP,DFBN,FO.RD
;
;       CREATE DATA SET FOR A FILE NAMED BLOCK.DAT
;
DSTP:   .WORD 0,0
        .WORD 0,0
        .WORD INAMSZ,INAM
INAM:   .ASCII /BLOCK.DAT/
INAMSZ=.-INAM
        .EVEN
;
;       CREATE DEFAULT FILE NAME BLOCK
;               -NAME IS BLOCK
;               -FILE TYPE IS DAT
;               -VERSION 1
;               -DEVICE DL (RL01)
;               -UNIT 1
```

```
;
DFBN:   NMBLK$ BLOCK,DAT,1,DL,1
        .EVEN
;
;       ESTABLISH I/O STATUS BLOCK
;
IOBK:   .WORD 0,0
        .END
```

C.3.3.2 Subroutine SLWDMP

```
        .TITLE SLWDMP
        .IDENT /FV01/
;
;**********************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT COORDINATES THE READING OF THE FILE CREATED TO PROVIDE
;       HIGH SPEED DATA TRANSFER RATES FROM THE LPA11-K TO AN RL01
;       DISK.
;
;       NOTE:   A FILE MUST HAVE BEEN WRITTEN USING THE FASOPN,FASDMP
;               ,FASCLS FAMILY OF SUBROUTINE IN ORDER FOR THIS FAMILY
;               OF SUBROUTINES TO WORK PROPERLY
;
;               THE SLWOPN,SLWDMP,SLWCLS FAMILY OF SUBROUTINES HAVE
;               SLOWER TRANSFER RATES THE THE FASOPN,FASDMP,FASCLS
;               FAMILY.
;
;       CALLING CONVENTIONS:
;
;       FORTRAN         CALL SLWDMP(BUF,N)
;
;       ARGUMENTS       BUF - THE INTEGER BUFFER TO BE TRANSFERED
;                       N   - THE NUMBER OF 32 WORD MULTIPLES OF
;                             THE DIMENSION OF BUF
;
;       NOTES:  THE SUBROUTINE SLWDMP READS WHOLE SECTORS FROM A
;               RL01 DISK (20. BLOCKS-10240. BYTES-5120. WORDS) USING
;               ONE INTERMEDIATE BUFFER OF THIS SIZE.
;
;               THE SIZE OF BUF MUST BE AN INTEGER MULTIPLE OF 5120.
;
;***********************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;**********************************************************************
;
        .GLOBL SLWDMP                   ; ADDRESS OF THE SWLDMP SUBROUTINE
        .GLOBL FDBIN                    ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL                   ; ADDRESS OF THE REGISTER SAVING SUBROUTINE
        .GLOBL VBN                      ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
        .GLOBL END                      ; ADDRESS OF THE END OF THE INTERMEDIATE
                                        ;   BUFFER ADDRESS
        .GLOBL START                    ; ADDRESS OF THE BEGINING OF THE INTER-
                                        ;   MEDIATE BUFFER
        .GLOBL LOC                      ; ADDRESS OF THE NEXT WORD  OF THE INTER-
                                        ;   MEDIATE BUFFER TO BE READ
        .GLOBL IOBK                     ; ADDRESS OF THE I/O STATUS BLOCK
```

```
;
;*****************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;*****************************************************************
;
        .MCALL READ$            ; BLOCK I/O READ MACRO
        .MCALL WAIT$            ; WAIT FOR I/O TRANSFER MACRO
;
;*****************************************************************
;
;       START OF SLWDMP CODE
;
;*****************************************************************
;
SLWDMP: JSR PC,$SAVAL           ; SAVE ALL OF THE REGISTERS
        MOV (R5)+,R0            ; GET THE NUMBER OF ARGUMENTS (UNNESSESARY)
        MOV (R5)+,R1            ; GET STARTING ADDRESS OF DATA BLOCK
        MOV @(R5)+,R2           ; GET THE SIZE OF THE DATA BLOCK IN 32.
                                ; WORD MULTILPLES
        MOV LOC,R3              ; PUSH THE LOCATION OF THE FIRST WORD OF
                                ; THE INTERMEDIATE BUFFER TO BE READ INTO
                                ; R3
        SUB #START,R3           ; TEST TO SEE IF WE ARE AT THE START OF THE
                                ; INTERMEDIATE BUFFER
        BEQ KERN                ; IF WE ARE, READ IN BUFFER FULL OF DATA FIRST
        SUB #10240.,R3          ; TEST TO SEE IF WE HAVE READ THROUGH THE
                                ; INTERMEDIATE BUFFER
        BGE KERN                ; IF WE HAVE, READ IN A NEW BUFFER FULL OF
                                ; DATA
        JMP TRANS               ; IF WE ARE SOMEWHERE IN THE MIDDLE OF THE
                                ; THE BUFFER TRANSFER DATA TO THE DATA BUFFER
;
;       READ DATA INTO THE INTERMEDIATE BUFFER
;               -FILE DESCRIPTOR BLOCK AT ADDRESS FDBIN
;               -INTERMEDIATE BUFFER STARTS AT ADDRESS START
;               -TRANSFER 10240. BYTES
;               -VIRTUAL BLOCK NUMBER AT ADDRESS VBN
;               -EVENT FLAG 23. IS USED FOR I/O COORDINATION
;               -I/O STATUS BLOCK IS AT ADDRESS IOBK
;
KERN:   READ$ #FDBIN,#START,#10240.,,#VBN,#23.,,#IOBK
;
;       WAIT FOR THE DATA TO BE TRANSFERED
;               -EVENT FLAG 23. IS USED FOR I/O COORDINATION
;               -I/O STATUS BLOCK IS AT ADDRESS IOBK
;
        WAIT$ #FDBIN,#23.,,#IOBK
        MOV #START,LOC          ; REPOSITION FIRST WORD OF THE
                                ; INTERMEDIATE BUFFER TO BE READ
                                ; TO THE START OF THE BUFFER
        ADD #20.,VBN+2          ; INCREMENT THE VIRTUAL BLOCK
                                ; NUMBER
;
;       TRANSFER THE DATA TO THE DATA BUFFER FROM THE INTERMEDIATE
;       BUFFER
;
TRANS:  MOV LOC,R3              ; GET THE ADDRESS OF THE FIRST FILE WORD
                                ; OF THE INTERMEDIATE BUFFER TO BE READ
AGAIN:  .REPT 32.               ; TRANSFER THE DATA
        MOV (R3)+,(R1)+
        .ENDR
        SOB R2,AGAIN
        MOV R3,LOC              ; SET THE NEXT WORD OF THE INTERMEDIATE
                                ; BUFFER TO BE READ TO ITS NEW VALUE
```

```
;
;               THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
START:  .BLKB 10240.                    ; RESERVE 10240.BYTES FOR THE INTERMEDIATE
                                        ; BUFFER
END:    .WORD 0                         ; RESERVE A WORD FOR THE END OF BUFFER
                                        ; ADDRESS
LOC:    .WORD 0                         ; RESERVE A WORD FOR THE ADDRESS OF THE
                                        ; NEXT WORD OF THE INTERMEDIATE BUFFER
                                        ; TO BE READ
VBN:    .BLKW 2                         ; RESERVE 2 WORDS FOR THE VIRTUAL
                                        ; BLOCK NUMBER
        .END
```

C.3.3.3 Subroutine SLWCLS

```
        .TITLE SLWCLS
        .IDENT /FV01/
;
;**********************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT CLOSES THE FILE OPENED BY SLWOPN.
;
;       ARGUMENTS       NONE
;
;**********************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;**********************************************************************
;
        .GLOBL SLWCLS                   ; ADDRESS OF THE SLWCLS SUBROUTINE
        .GLOBL FDBIN                    ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL                   ; ADDRESS OF THE REGISTER SAVING ROUTINE
;
;**********************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;**********************************************************************
;
        .MCALL CLOSE$                   ; CLOSE MACRO
;
;**********************************************************************
;
;       START OF THE SLWCLS CODE
;
;**********************************************************************
;
SLWCLS: JSR PC,$SAVAL                   ; SAVE ALL REGISTERS
        CLOSE$ #FDBIN                   ; CLOSE THE FILE POINTED TO BY THE FILE
                                        ; DESCRIPTOR BLOCK AT ADDRESS FDBIN
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
        .END
```

C.3.3.4 Subroutine SMLOPN

```
        .TITLE SMLOPN
        .IDENT /FV01/
;
;**************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT OPENS THE LARGE CONTIGUOUS FILE ON LOGICAL UNIT 8.
;       THAT WAS CREATFD BY FASOPN AND FILLED BY FASDMP.  THIS
;       SUBROUTINE (ALONG WITH ITS SIBLINGS ) IS USED TO READ
;       THE BLOCK STRUCTURED FILE THAT WAS CREATED FOR HIGH
;       SPEED DATA TRANSFER FROM A LPA11-K TO A RL01 OF LARGE
;       QUANTITIES OF CONTIUGOUS DATA.
;
;       NOTE:   A FILE MUST HAVE BEEN CREATED BY THE FASOPN,
;               FASDMP,FASCLS FAMILY OF SUBROUTINES IN ORDER
;               FOR THIS FAMILY OF SUBROUTINES TO WORK.
;
;               DO NOT USE EVENT FLAG 23(DECIMAL) FOR ANY OTHER
;               APPLICATION, IT IS USED TO COORDINATE I/O.
;
;               THIS IS A SMALL VERSION OF SLWOPN
;
;**************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;**************************************************************
;
        .GLOBL SMLOPN           ; ADDRESS OF THE SMLOPN SUBROUTINE
        .GLOBL FDBIN            ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL           ; ADDRESS OF REGISTER SAVING SUBROUTINE
        .GLOBL VBN              ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
        .GLOBL END              ; ADDRESS OF THE END OF THE INTERMEDIATE
                                ;   BUFFER ADDRESS
        .GLOBL START            ; ADDRESS OF THE BEGINING OF THE INTER-
                                ;   MEDIATE BUFFER
        .GLOBL LOC              ; ADDRESS OF THE NEXT FREE LOCATION IN
                                ;   THE INTERMEDIATE BUFFER
        .GLOBL DFBN             ; ADDRESS OF THE DEFAULT NAME BLOCK
        .GLOBL DSTP             ; ADDRESS OF THE DATA SET DESCRIPTOR
        .GLOBL IIOBK            ; ADDRESS OF THE I/O STATUS BLOCK
;
;****************************************************************
;
;       MACRO CALL DEFINITIONS (ASSEMBLY-TIME)
;
;****************************************************************
;
        .MCALL FDBDF$           ; ALLOCATE FILE DESCRIPTOR MACRO
        .MCALL FDAT$A           ; INITIALIZE FILE ATTRIBUTE MACRO
        .MCALL FDRC$A           ; INITIALIZE RECORD ACCESS SECTION MACRO
        .MCALL FDBK$A           ; INITIALIZE BLOCK ACCESS SECTION MACRO
        .MCALL FDOP$A           ; INITIALIZE FILE-OPEN SECTION MACRO
        .MCALL NMBLK$           ; DEFAULT FILE NAME MACRO
;
;****************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;****************************************************************
;
        .MCALL OPEN$R           ; OPEN TO READ BLOCKS MACRO
        .MCALL FDBK$R           ; INITIALIZE BLOCK ACCESS SECTION MACRO
;
;****************************************************************
```

```
;
;       START OF SMLOPN CODE
;
;*************************************************************************
;
SMLOPN: JSR PC,$SAVAL           ; SAVE ALL OF THE REGISTERS
        MOV #END,END            ; STORE THE ADDRESS OF THE END OF THE
        SUB #2,END              ; INTERMEDIATE BUFFER IN END
        MOV #START,LOC          ; STORE THE ADDRESS OF THE BEGINING OF
                                ; THE INTERMEDIATE BUFFER IN LOC
        CLR VBN                 ; INITIALIZE THE VIRTUAL BLOCK NUMBER
        MOV #1,VBN+2            ; TO 1(DECIMAL)
;
;       OPEN THE FILE ON LOGICAL UNIT 8. FOR BLOCK I/O
;
        OPEN$R #FDBIN,#8.,#DSTP,#FD.RWM
;
;       THIS STEP IS REQUIRED TO INITIALIZE THE VIRTUAL BLOCK NUMBER
;       REGION OF THE FILE DESCRIPTOR BLOCK (FOR BLOCK I/O ONLY)
;
        FDBK$R #FDBIN,#START,#2048.,#VBN,#23.
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
;
;**********************************************************************
;
;       THE FOLLOWING SECTION IS REQUIRED TO ESTABLISH THE FILE
;       DESCRIPTOR BLOCK
;
;**********************************************************************
;
;       ALLOCATE FILE DESCRIPTOR BLOCK STARTING AT FDBIN
;
FDBIN:  FDBDF$
;
;       INITIALIZE FILE ATTRIBUTES SECTION OF THE FDB
;               -FIXED RECORDS
;               -RECORD IS TO BE PRECEDED BY <LF> AND FOLLOWED BY <CR>
;               -RECORD SIZE OF 10240. BYTES
;               -INITIAL SIZE DEFAULTED TO FILE SIZE
;               -NO EXTENTIONS ON AN OPEN FOR READ
;
        FDAT$A R.FIX,FD.CR,10240.
;
;       INITIALIZE RECORD ACCESS SECTION OF THE FDB FOR BLOCK I/O
;
        FDRC$A FD.RWM
;
;       INITIALIZE BLOCK ACCESS SECTION OF THE FDB
;               -BUFFER FOR BLOCK I/O STARTS AT ADDRESS START
;               -BUFFER FOR BLOCK I/O IS 2048. BYTES LONG
;               -THIRD ARGUMENT IS A DUMMY ARGUMENT  ( FOR COMPATABILITY
;                       WITH FDBK$R )
;               -EVENT FLAG 23. IS TO BE USED FOR I/O COORDINATION
;               -I/O STATUS BLOCK IS AT ADDRESS IIOBK
;
        FDBK$A START,2048.,,23.,IIOBK
;
;       INITIALIZE FILE OPEN SECTION OF THE FDB
;               -LOGICAL UNIT NUMBER IS 8.
;               -DATA SET POINTER ADDRESS IS DSTP
;               -DEFAULT FILE NAME BLOCK ADDRESS IS DFBN
;               -AN OLD FILE IS TO BE OPENED FOR READING
;
```

```
        FDOP$A 8.,DSTP,DFBN,FO.RD
;
;       CREATE DATA SET FOR A FILE NAMED BLOCK.DAT
;
DSTP:   .WORD 0,0
        .WORD 0,0
        .WORD INAMSZ,INAM
INAM:   .ASCII /BLOCK.DAT/
INAMSZ=.-INAM
        .EVEN
;
;       CREATE DEFAULT FILE NAME BLOCK
;               -NAME IS BLOCK
;               -FILE TYPE IS DAT
;               -VERSION 1
;               -DEVICE DL (RL01)
;               -UNIT 1
;
DFBN:   NMBLK$ BLOCK,DAT,1,DL,1
        .EVEN
;
;       ESTABLISH I/O STATUS BLOCK
;
IIOBK:  .WORD 0,0
        .END
```

C.3.3.5 Subroutine SMLDMP

```
        .TITLE SMLDMP
        .IDENT /FV01/
;
;**************************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT COORDINATES THE READING OF THE FILE CREATED TO PROVIDE
;       HIGH SPEED DATA TRANSFER RATES FROM THE LPA11-K TO AN RL01
;       DISK.
;
;       NOTE:   A FILE MUST HAVE BEEN WRITTEN USING THE FASOPN,FASDMP
;               ,FASCLS FAMILY OF SUBROUTINE IN ORDER FOR THIS FAMILY
;               OF SUBROUTINES TO WORK PROPERLY
;
;               THE SMLOPN,SMLDMP,SMLCLS FAMILY OF SUBROUTINES HAVE
;               SLOWER TRANSFER RATES THE THE FASOPN,FASDMP,FASCLS
;               FAMILY.
;
;       CALLING CONVENTIONS:
;
;       FORTRAN             CALL SMLDMP(BUF,N)
;
;       ARGUMENTS           BUF - THE INTEGER BUFFER TO BE TRANSFERED
;                           N   - THE NUMBER OF 32 WORD MULTIPLES OF
;                                 THE DIMENSION OF BUF
;
;       NOTES:  THE SUBROUTINE SMLDMP READS WHOLE SECTORS FROM A
;               RL01 DISK (20. BLOCKS-10240. BYTES-5120. WORDS) USING
;               ONE INTERMEDIATE BUFFER OF THIS SIZE.
;
;               THE SIZE OF BUF MUST BE AN INTEGER MULTIPLE OF 5120.
;
;***************************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;**************************************************************************
;
```

```
          .GLOBL SMLDMP               ; ADDRESS OF THE SMLDMP SUBROUTINE
          .GLOBL FDBIN                ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
          .GLOBL $SAVAL               ; ADDRESS OF THE REGISTER SAVING SUBROUTINE
          .GLOBL VBN                  ; ADDRESS OF THE VIRTUAL BLOCK NUMBER
          .GLOBL END                  ; ADDRESS OF THE END OF THE INTERMEDIATE
                                      ;    BUFFER ADDRESS
          .GLOBL START                ; ADDRESS OF THE BEGINING OF THE INTER-
                                      ;    MEDIATE BUFFER
          .GLOBL LOC                  ; ADDRESS OF THE NEXT WORD  OF THE INTER-
                                      ;    MEDIATE BUFFER TO BE READ
          .GLOBL IIOBK                ; ADDRESS OF THE I/O STATUS BLOCK
;
;**********************************************************************
;
;         MACRO CALL DEFINITIONS (RUN-TIME)
;
;**********************************************************************
;
          .MCALL READ$                ; BLOCK I/O READ MACRO
          .MCALL WAIT$                ; WAIT FOR I/O TRANSFER MACRO
;
;**********************************************************************
;
;         START OF SMLDMP CODE
;
;**********************************************************************
;
SMLDMP:   JSR R5,$SAVAL               ; SAVE CONTENTS OF THE REGISTERS
          MOV (R5)+,R0                ; GET THE NUMBER OF ARGUMENTS (SHOULD BE 2)
          MOV (R5)+,R1                ; GET STARTING ADDRESS OF DATA BUFFER
          MOV @(R5)+,R2               ; GET THE SIZE OF THE DATA BUFFER IN 2.,
                                      ; WORD MULTIPLES
          MOV LOC,R3                  ; PUSH THE LOCATION OF THE FIRST WORD OF
                                      ; THE INTERMEDIATE BUFFER TO BE READ INTO
                                      ; R3
          SUB #START,R3               ; TEST TO SEE IF WE ARE AT THE START OF THE
                                      ; INTERMEDIATE BUFFER
          BEQ KERN                    ; IF WE ARE, READ IN BUFFER FULL OF DATA FIRST
          SUB #2048.,R3               ; TEST TO SEE IF WE HAVE READ THROUGH THE
                                      ; INTERMEDIATE BUFFER
          BGE KERN                    ; IF WE HAVE, READ IN A NEW BUFFER FULL OF
                                      ; DATA
          JMP TRANS                   ; IF WE ARE SOMEWHERE IN THE MIDDLE OF THE
                                      ; THE BUFFER TRANSFER DATA TO THE DATA BUFFER
;
;         READ DATA INTO THE INTERMEDIATE BUFFER
;                 -FILE DESCRIPTOR BLOCK AT ADDRESS FDBIN
;                 -INTERMEDIATE BUFFER STARTS AT ADDRESS START
;                 -TRANSFER 2048. BYTES
;                 -VIRTUAL BLOCK NUMBER AT ADDRESS VBN
;                 -EVENT FLAG 23. IS USED FOR I/O COORDINATION
;                 -I/O STATUS BLOCK IS AT ADDRESS IIOBK
;
KERN:     READ$ #FDBIN,#START,#2048.,#VBN,#23.,#IIOBK
;
;         WAIT FOR THE DATA TO BE TRANSFERED
;                 -EVENT FLAG 23. IS USED FOR I/O COORDINATION
;                 -I/O STATUS BLOCK IS AT ADDRESS IIOBK
;
          WAIT$ #FDBIN,#23.,#IIOBK
          MOV #START,LOC              ; REPOSITION FIRST WORD OF THE
                                      ; INTERMEDIATE BUFFER TO BE READ
                                      ; TO THE START OF THE BUFFER
          ADD #4.,VBN+2               ; INCREMENT THE VIRTUAL BLOCK
                                      ; NUMBER
```

```
;
;       TRANSFER THE DATA TO THE DATA BUFFER FROM THE INTERMEDIATE
;       BUFFER
;
TRANS:  MOV LOC,R3              ; GET THE ADDRESS OF THE FIRST FREE WORD
                                ; OF THE INTERMEDIATE BUFFER TO BE READ
AGAIN:  .REPT 32.               ; TRANSFER THE DATA
        MOV (R3)+,(R1)+
        .ENDR
        SOB R2,AGAIN
        MOV R3,LOC              ; SET THE NEXT WORD OF THE INTERMEDIATE
                                ; BUFFER TO BE READ TO ITS NEW VALUE
;
;               THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
START:  .BLKB 2048.             ; RESERVE 10240.BYTES FOR THE INTERMEDIATE
                                ; BUFFER
END:    .WORD 0                 ; RESERVE A WORD FOR THE END OF BUFFER
                                ; ADDRESS
LOC:    .WORD 0                 ; RESERVE A WORD FOR THE ADDRESS OF THE
                                ; NEXT WORD OF THE INTERMEDIATE BUFFER
                                ; TO BE READ
VBN:    .BLKW 2                 ; RESERVE 2 WORDS FOR THE VIRTUAL
                                ; BLOCK NUMBER
        .END
```

C.3.3.6 Subroutine SMLCLS

```
        .TITLE SMLCLS
        .IDENT /FV01/
;
;***********************************************************************
;
;       THIS SUBROUTINE IS EITHER FORTRAN OR MACRO CALLABLE.
;       IT CLOSES THE FILE OPENED BY SMLOPN.
;
;       ARGUMENTS       NONE
;
;***********************************************************************
;
;       GLOBAL SYMBOL DEFINITIONS
;
;***********************************************************************
;
        .GLOBL SMLCLS           ; ADDRESS OF THE SMLCLS SUBROUTINE
        .GLOBL FDBIN            ; ADDRESS OF THE FILE DESCRIPTOR BLOCK
        .GLOBL $SAVAL           ; ADDRESS OF THE REGISTER SAVING ROUTINE
;
;***********************************************************************
;
;       MACRO CALL DEFINITIONS (RUN-TIME)
;
;***********************************************************************
;
        .MCALL CLOSE$           ; CLOSE MACRO
;
;***********************************************************************
;
;       START OF THE SMLCLS CODE
;
;***********************************************************************
;
SMLCLS: JSR PC,$SAVAL           ; SAVE ALL REGISTERS
        CLOSE$ #FDBIN           ; CLOSE THE FILE POINTED TO BY THE FILE
                                ; DESCRIPTOR BLOCK AT ADDRESS FDBIN
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
        .END
```

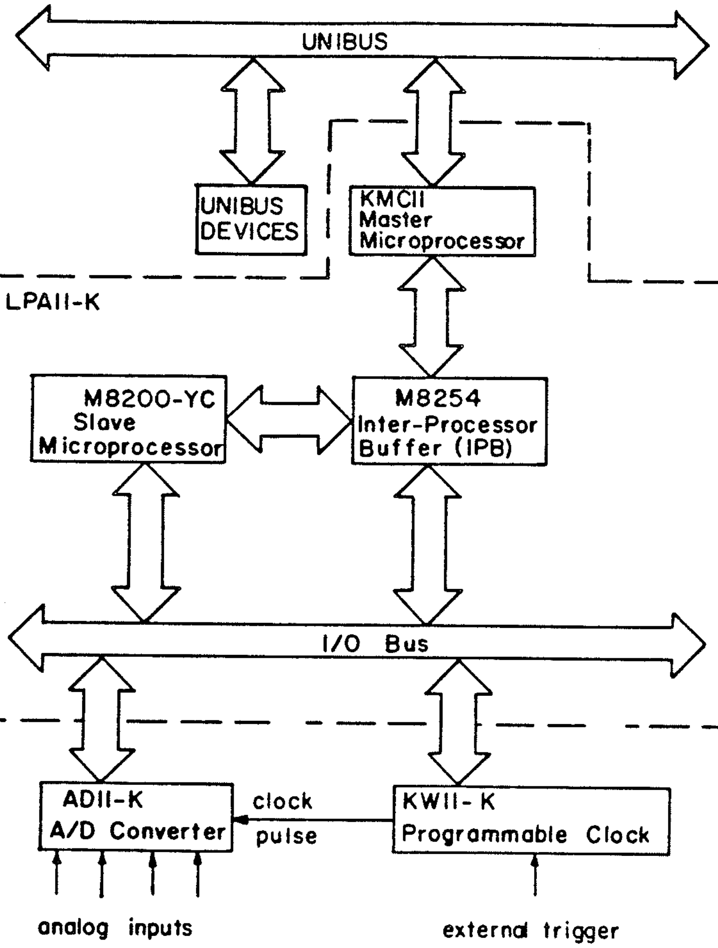


Figure C.1: The LPA-11K and laboratory devices data path schematic.

Appendix D

Single Point Calculations

D.1 CALCULATION OF THE MEAN INPUT VALUE

The input signal was not AC coupled electronically because it was known to be skewed. By retaining a DC bias the optimum digitization accuracy could be used, the A/D converter could resolve the signal into 2.44 mV steps from -5 V to 5 V. Thus the digital signal had a mean offset which had to be eliminated before most subsequent processing.

As the PDP-11 architecture uses a two's-complement 16 bit integer arithmetic, the maximum value a number can have is 32676. This is true even for data that has been declared as integer longwords in FORTRAN as the DEC [1979b] implementation does not use the 32 bits. A special subroutine was developed to accumulate large sums in a two word integer array. This subroutine is listed as FASSUM. The basic idea was to load -32676 into a register and add the elements of an input data buffer to it until the contents changed sign. A change in sign was easy to test for. When the sign change occured one was added to a second register (initially zero) and 32676 was subtracted off of the first register. Then input data was added to the register until either the process repeated or ran out of data. When it was finished 32676 was added to the register and the values were returned.

The subroutine could only handle up to 5120 data points so intermediate accumulations had to be made for larger data sets. From this accumulated mean an integer offset could be calculated. A MACRO subroutine, FASAC, could then be used to very quickly AC couple (digitally) the signal.

It sould be noted that because of data conversion problems nearly all signal analysis subroutines had to be written in MACRO. The accumulation of the sum of all of the values using FORTRAN took 6 minutes for 2,000,000 samples, using the FASSUM routine this was done in about 20 seconds.

D.2 THE CENTRAL MOMENTS

The central moments were computed using a MACRO subroutine FASSTS. This routine used the floating point processor to calculate all of the moments up to 6th order in double precision. The moments were corrected for any residual zero error using a correction subroutine STSCOR even though the input stream had been digitally AC coupled. For a sine wave input the flatness factor was found to be within 0.1% of its theoretical value of 1.5 when using these subroutines.

D.3 THE SPECTRA

The spectra were calculated using the FFT and spectra subroutines supplied by DEC [1981c]. Unfortunately this did not give a sufficiently wide bandwidth. So a 'high frequency' spectra was calculated using the data as is. Then after storing this spectra a 'low frequency' spectra was calculated. To do this the data stream was low-pass filtered using a second order Butterworth IIR (infinite impulse response) filter. These filters are described in Beauchamp and Yuen [1979] and the subroutines were FILTER, SETFLT and LPFILT. The cut-off frequency was set to 1/128 of the sampling frequency. Then the data was 'sampled' at 1/64 of the sampling rate and the spectra calculated. The low-frequency spectra had to be multiplied by 64 to make them match the high-frequency spectra in the overlap range, because of the bandwidth shift. The low-frequency spectra suffered from poor averaging because of the lower frequencies and reduced sample size. The low-frequency spectra were smoothed using four passes of a four point moving average. A Hamming window was used on all input data for spectral calculations.

D.4 PROBABILITY FUNCTIONS

It was found that the probability distribution and density functions could be computed very easily and swiftly with FORTRAN subroutines. It was observed that as the input stream was comprised of integers ranging from 0 to 4096 array element locations could be computed from them. Thus if the input value was divided by 4 (integer calculation) and had 1 added to it, it could refer to one element of a 1024 element array. This would correspond to a 10 mV bin in a 10 V peak to peak signal. Using this address the number of occurances of the element could easily be counted. Note that no AC coupling should be done. From this histogram it is very easy to calculate the probability distribution function and then differentiate this function to obtain the probability density function. The subroutine that did the histograming was PHIST, the probability distribution function was PDENS and the probability density was PDF.

D.5 AUTOCORRELATION

The auto correlation was computed directly using 56 quasi-logrithmically distributed delays. The maximum time delay corresponded to 5120 sampling periods. Prior to processing the data the input data stream was digitally AC coupled, but the autocorrelations were corrected for residual DC bias. The residual DC bias was calculated using a MACRO subroutine MEAN which calculated the DC part (after AC coupling) using double precision floating point arithmetic.

The correlations were done using the subroutine PTCOR, a MACRO subroutine, which computed the correlation between two data streams in double precision floating point. The array used to store the correlation input stream had to be twice the size of the maximum delay. Then by making calls to PTCOR with the start address of one data stream at the begining of the array and the address of the second stream shifted back, the delayed correlation could be calculated. After all the correlations were calculated the last half of the array was shifted to the beginning using a high speed memory to memory data transfer subroutine BLKSFT, and new data written into the free part of the array. This process continued until the end of the data.

D.6 ZARIC'S CONDITIONAL SAMPLING

Zaric's conditional sampling routine required the differentiated signal values. To obtain them a broadband FIR (finite impulse response) digital filter was written, DIFTOR, which was based on the one given by Chen [1979]. This differentiation produced a delayed output stream, and in order to maintain concurrency for forming the appropriate products the data stream had to be delayed as well. This was done in the subroutine TDELAY.

To set the conditioning parameters, an initial pass through the data stream had to be made. In this pass the data that had to be accumulated to set the test parameters was calculated in ZSET. The data calculated in ZSET was then used in TSTSET to establish the conditioning parameter.

At this point two comments should be made. First, the output from ZSET depends on the value of IMP used for accumulating the results. IMP was set to the most probable shear stress, this point was very difficult to establish accurately and it is very arbitrary. Second the test conditions used several arbitrary constants and it is uncertain how much influence they could have on any physical picture that may be drawn.

After the conditioning parameters were set a second pass on the data stream was made. In this pass the test parameters were gated in subroutine GATE to produce an output stream of -1 for an ejection, 1 for a sweep and 0 for neither event. This output gating stream and the data stream was used to produce the probability histogram in ZHIST. The resulting histogram was then used as input for the subroutine PDFMOM which calculated the mean value and the central moments up to the 4th order for the total signal and each of the three events.

D.7 COMPUTER CODE

D.7.1 The single channel analysis program

This program was used to compute the first six central moments, the spectrum and the probability density function of a single channel input stream, The data was assumed to have been collected by the previously described data acquisition program.

```
C-------------------------------------------------------------------------------
C
C       This program processes the data collected from a single channel of the
C       LPA-11K.  It accepts sampling rates up to the maximum that the A/D
C       converter can process.  The data is assumed to have been written on
C       disk drive 01 by the program FAST, which creates the all of the
C       necsesary files on this disk.
C
C       The quantities that are computed are:
C                     1)  the mean off-set
C                     2)  the RMS input voltage
C                     3)  the skewness
C                     4)  the kurtosis (flatness factor)
C                     5)  the super-skewness
C                     6)  the super-flatness
C                     7)  the spectrum (over a 4.8 decade range)
C                     8)  the probability distribution
C                     9)  the probability density function.
C
C       The processing time for 2400 buffers of data is roughly 2.5 hours.
C
C-------------------------------------------------------------------------------
C
C       Data declaration and definition region
C
C-------------------------------------------------------------------------------
C
        INTEGER IREAL(1024),IMAG(1024),K1(2),IOFF
C
C       IREAL    - a 1024 word buffer used to store the real part of the signal
C                  or the spectrum
C       IMAG     - a 1024 word buffer used to store the imaginary part of the
C                  signal of the spectrum
C       K1       - a 2 word buffer used to compute the integral (sum) of the
C                  signal
C       IOFF     - the integer value of the offset
C
        INTEGER IERROR,ISCALE,ITAPE,ICHN,IBAND,ISIZE,I,J,K,L,N
C
C       IERROR   - an error parameter used by the FFT subroutine
C       ISCALE   - the scaling parameter returned by the FFT subroutine
C       ITAPE    - the tape number, used to identify the data
C       ICHN     - the channel number, used to identify the data
C       IBAND    - the band number, used to identify the data
C       ISIZE    - the number of 1024 word buffers of data that were collected
C       I,J,K,L  - dummy variables
C       and N
C
        INTEGER A(4),B(5)
C
C       A        - a 4 word buffer used to store the ascii code for the time
C       B        - a 5 word buffer used to store the ascii code for the date
C
        REAL TEMP1(1024),TEMP2(1024),MEAN,F,DT,DCPART,ACPART
C
C       TEMP1    - a 1024(X2) word buffer used to store the spectrum,
C                  autocorrelation and probability distribution
C       TEMP2    - a 1024(X2) word buffer used to compute the spectrum,
C                  autocorrelation and to store the probability density
C                  function
C       MEAN     - the real value of the mean input voltage
C       F        - the cut-off frequency of the low-pass filter used to
C                  extend the spectrum calculations
C       DT       - the time interval between successive samples
C       DCPART   - the DC part of the initial signal (m/sec,etc)
```

```
C       ACPART  - the AC part of the initial signal (m/sec,etc)
C
        REAL TO,T1,T2,FACT,MIN,NEXT,POWER
C
C       TO      - the initial time after midnight in seconds
C       T1,T2   - the elapsed time counters
C       FACT    - a temporary multiplying factor
C       MIN     - the minimum voltage considered for the probability
C                 distribution and probability density calulations,
C                 incremented for writing out these values
C       NEXT    - a temporary storage value
C       POWER   - the integral of the high frequency power spectrum
C
        REAL*8 STATS(6),H(6)
C
C       STATS   - a 6(x4) word buffer used to store the moments
C       H       - a 6(X4) word buffer used to accumulate the moments
C
C-----------------------------------------------------------------------
C
C       Subroutine declarations
C
C-----------------------------------------------------------------------
C
        INTEGER SLWOPN,SLWDMP,SLWCLS
C
C       SLWOPN  - the subroutine that opens the data file on disk 01
C       SLWDMP  - the subroutine that reads the data file
C       SLFCLS  - the subroutine that closes the data file
C
        INTEGER FASSUM,FASAC,FASSTS,STSCOR
C
C       FASSUM  - the subroutine that integrates the data signal
C       FASAC   - the subroutine that AC couples the signal
C       FASSTS  - the subroutine that computes the moments
C       STSCOR  - the subroutine that corrects the moments for residual offset
C
        INTEGER FILTER,LPFILT
C
C       FILTER  - the subroutine that computes the filter constants and
C                 sets the filter LPFILT
C       LPFILT  - the subroutine that low-pass filters the signal,
C                 the role of rate is 12 db/oct
C
        INTEGER PHIST,PDENS,PDF
C
C       PHIST   - the subroutine that computes the initial probability
C                 distribution
C       PDENS   - the subroutine that computes the final probability
C                 distribution
C       PDF     - the subroutine that computes the probability density
C                 function from the probability distribution
C
        INTEGER TIME,DATE,FFT,POWRSP,IFIX
C
C       TIME    - the subroutine that returns the ascii code for the time
C       DATE    - the subroutine that returns the ascii code for the date
C       FFT     - the subroutine that computes the fast Fourier transform of
C                 the input signal
C       POWRSP  - the subroutine that computes the power spectrum from the
C                 fast Fourier transform
C       IFIX    - the subroutine that converts reals to integers
C
        REAL RINTGL,SECNDS,FLOAT
C
C       RINTGL  - a subroutine that computes the integral of a real input
```

```
C                    function of raw data in the form X,Y , where X and Y are
C                    arrays
C       SECNDS   - a subroutine that computes the time elapsed from some initial
C                    time
C       FLOAT    - the subroutine that converts integers to reals
C
C-----------------------------------------------------------------------
C
C       Start of the code
C-----------------------------------------------------------------------
C
C       get the intial time
C
        T0=SECNDS(0.0)
        CALL TIME(A)
        CALL DATE(B)
C
C       type the intial time and the present date to the terminal that ran the
C       program
C
        TYPE 1000,(A(I),I=1,4),(B(I),I=1,5)
1000    FORMAT(' ',80('*')//'0','Data processing commenced at ',4A2,' on '
     *  ,5A2/' ')
C
C----------------------------------------------------------------------
C
C       this section gets the digitization and signal information that was
C       saved by the program FAST
C
C----------------------------------------------------------------------
C
C       open the information file that was written by FAST on disk DL01
C
C       Note; logical unit number 3 must be assigned to DL1: when the
C             program is being Task Built. Use
C               ASN=DL1:3        in the options section of TKB
C
        OPEN(UNIT=3,TYPE='OLD',NAME='DL1:INFO.DAT')
C
C       read in the number of channels sampled,tape number and
C       the band number of the input signal
C
        READ(3,*) I,ITAPE,IBAND
C
C       if I is not equal to 1 type a message and halt
C
        IF(I.NE.1) GO TO 130
C
C       read in the tape channel number
C
        READ(3,*) ICHN
C
C       read in the sampling interval
C
        READ(3,*) DT
C
C       read in the DC and AC parts of the signal
C
        READ(3,*) DCPART,ACPART
C
C       read in the number of 1024 word blocks
C
        READ(3,*) ISIZE
C
C       close the information file
C
```

```
        CLOSE(UNIT=3)
C
C       type the tape, channel and band numbers,and the number
C       of 1024 word blocks of data on the terminal
C
        TYPE 1001,ITAPE,ICHN,IBAND,ISIZE
1001    FORMAT(' ','For tape ',I2,' channel ',I2,' band ',I2
     *   /' ','With ',I4,' buffers (1024 words) of data')
C
C       type the sampling interval and the mean and RMS values of the
C       signal on the terminal
C
        TYPE 1002,DT,DCPART,ACPART
1002    FORMAT(' ','The input data was sampled every',3X,E10.3,' sec'/' '
     * ,'The mean of the input signal was ',F8.3,1X,/' '
     * ,'The RMS of the input signal was  ',F8.3,1X,/'0')
C
C-----------------------------------------------------------------------
C
C       this section opens the print file named PRINT.TXT
C
C-----------------------------------------------------------------------
C
        OPEN(UNIT=2,NAME='PRINT.TXT',TYPE='NEW')
C
C       write in the lead information to the print file
C
        WRITE(2,2000)
2000    FORMAT('1')
        WRITE(2,1000) (A(I),I=1,4),(B(I),I=1,5)
        WRITE(2,1001) ITAPE,ICHN,IBAND,ISIZE
        WRITE(2,1002) DT,DCPART,ACPART
C
C-----------------------------------------------------------------------
C
C       this section computes the integer offset on the signal
C
C-----------------------------------------------------------------------
C
C       open the data file to compute the offset
C
        CALL SLWOPN
C
C       initialize the mean value
C
        MEAN=0.0
C
C       integrate the signal to get the offset.
C
        DO 100 I=1,ISIZE
C
C       fill ireal with 32X32 words of data
C
        CALL SLWDMP(IREAL,32)
C
C       calculate the intermediate sum
C
        CALL FASSUM(IREAL,1024,K1)
C
C       accumulate the mean value
C
        MEAN=MEAN+32767.0*FLOAT(K1(1))+FLOAT(K1(2))
100     CONTINUE
C
C       calculate the mean input value
C
```

```
      MEAN=MEAN/(FLOAT(ISIZE)*1024.0)
C
C     compute the integer offset
C
      IOFF=IFIX(MEAN)
C
C     convert the mean input value to volts
C
      MEAN=(MEAN-2048.0)*2.44E-3
C
C     obtain the elapsed time
C
      T1=SECNDS(T0)
C
C     type the mean,offset and elapsed time on the terminal
C
      TYPE 1003,T1,MEAN,IOFF
1003  FORMAT(' ','Elapsed time',3X,F5.2,2X,'(sec)'/' ','Mean voltage'
     * ,2X,F6.3,2X,'(Volts)'/' ','Offset value',2X,I5/' ')
C
C     get the present time
C
      CALL TIME(A)
C
C     type the time onto the terminal
C
      TYPE 1004,(A(I),I=1,4)
1004  FORMAT(' ','The present time is ',4A2/' ')
C
C-----------------------------------------------------------------------
C
C     this section computes the moments of the signal
C
C-----------------------------------------------------------------------
C
C     get the number of seconds since midnight
C
      T1=SECNDS(0.0)
C
C     reset to read the file from the start
C
      CALL SLWCLS
      CALL SLWOPN
C
C     initialize the moments array
C
      DO 101 I=1,6
      STATS(I)=0.0
101   CONTINUE
C
C     accumulate over the whole data set
C
      DO 103 I=1,ISIZE
C
C     read in the buffer
C
      CALL SLWDMP(IREAL,32)
C
C     AC couple the signal
C
      CALL FASAC(IREAL,1024,IOFF)
C
C     compute the moments in the intermediate array H
C
      CALL FASSTS(IREAL,1024,H)
C
```

```
C       add the values in H to the values in STATS
C
        DO 102 J=1,6
        STATS(J)=STATS(J)+H(J)
102     CONTINUE
103     CONTINUE
C
C       average the moments
C
        DO 104 I=1,6
        STATS(I)=STATS(I)/FLOAT(ISIZE)
104     CONTINUE
C
C       correct the moments for offset error and non-dimensionalize the
C       third and higher moments
C
        CALL STSCOR(STATS)
C
C       get the elapsed time
C
        T2=SECNDS(T1)
C
C       convert the elapsed time to minutes
C
        T2=T2/60.0
C
C       type the elapsed time on the terminal
C
        TYPE 1005,T2
1005    FORMAT(' ','Elapsed time',T20,F7.3,2X,'(min)')
C
C       type the moments on to the terminal
C
        TYPE 1006,(STATS(I),I=2,6)
1006    FORMAT(' ','RMS',T20,F7.3,2X,'(Volts)'/' ','Skewness',T20,F7.3
     * /' ','Flatness',T20,F7.3/' ','Super-skewness',T20,F7.3/' ',
     * 'super-flatness',T20,F7.3/' ')
C
C       write the moments out to the print file
C
        WRITE(2,1006) (STATS(I),I=2,6)
C
C------------------------------------------------------------------------------
C
C       this section computes the spectrum of the signal
C
C------------------------------------------------------------------------------
C
C       write the heading for the high-frequency spectrum out to the
C       print file
C
        WRITE(2,2001)
2001    FORMAT('1',80('*')/'0',
     *  'The high-frequency part of the spectrum'/'0')
C
C       get the present time and type it on the terminal
C
        CALL TIME(A)
        TYPE 1004,(A(I),I=1,4)
C
C       reset to read the file from the start
C
        CALL SLWCLS
        CALL SLWOPN
C
C       start computation of the high frequency end of the spectrum
```

```
C
C
C       zero the buffer TEMP2 for accumulation of the power spectrum
C
        DO 105 I=1,1024
        TEMP2(I)=0.0
105     CONTINUE
C
C       do the calculations over the whole data set
C
        DO 109 I=1,ISIZE
C
C       zero the imaginary part of the signal
C
        DO 106 J=1,1024
        IMAG(J)=0
106     CONTINUE
C
C       read in the real part of the signal
C
        CALL SLWDMP(IREAL,32)
C
C       AC couple the signal
C
        CALL FASAC(IREAL,1024,IOFF)
C
C       window the data record
C
        CALL HAN(IREAL,1024,1024)
C
C       compute the fast Fourier transform
C
        CALL FFT(IERROR,1024,IREAL,IMAG,0,ISCALE)
C
C       do error handling
C
        IF(IERROR.EQ.0) GO TO 107
C
C       if there is an error indication type a message on the terminal
C
        TYPE 1007,IERROR
1007    FORMAT(' ','The FFT parameter IERROR is',2X,I3/' ')
107     CONTINUE
C
C       compute the power spectrum
C
        CALL POWRSP(1024,IREAL,IMAG,TEMP1)
C
C       compute the scaling factor
C
        FACT=2.0**(ISCALE*2)
C
C       accumulate the spectrum
C
        DO 108 J=1,1024
        TEMP2(J)=TEMP2(J)+FACT*TEMP1(J)
108     CONTINUE
109     CONTINUE
C
C       average the power spectrum for possitive frequencies
C
        N=1024
C
C       compute the relational scaling factor
C
C       FACT=[{2.44E-3(mV)/(2pi*buffer size)}**2]/[number of spectrum averaged]
```

```
C
         FACT=1.438202E-13/FLOAT(ISIZE)
         DO 110 I=2,512
         TEMP2(I)=(TEMP2(I)+TEMP2(N))
         TEMP2(I)=FACT*TEMP2(I)
         N=N-1
110      CONTINUE
         TEMP2(1)=2.0*FACT*TEMP2(1)
         TEMP2(513)=2.0*FACT*TEMP2(513)
C
C        fill TEMP1 with the frequencies that correspond to the power
C        spectrum in TEMP2
C
C        compute the frequency interval
C
         FACT=1.0/(1024.0*DT)
C
C        convert the frequency interval to kHz
C
         FACT=FACT/1000.0
C
C        set the initial value to 0
C
         TEMP1(1)=0.0
C
C
C        set the remaining values
C
         DO 111 I=2,513
         TEMP1(I)=FACT*FLOAT(I-1)
111      CONTINUE
C
C        write out the power spectrum
C
         WRITE(2,2002)
2002     FORMAT('0',80('*')//'0',4(1X,'Frequency',2X,'Spectrum',1X)//' '
     *   ,4(2X,'(kHz)',5X,'(pow/Hz)',1X)/' ')
         WRITE(2,2003) (TEMP1(I),TEMP2(I),I=2,513)
2003     FORMAT(' ',T1,4(2X,F6.2,3X,E10.3))
         WRITE(2,2004) TEMP2(1)
2004     FORMAT('0','The zero frequency component',2X,E10.3,2X,
     *   '(pow/Hz)'/'0',80('*'))
C
C        write the spectrum out to the plot file
C
         OPEN(UNIT=1,NAME='DL1:SNGPLT.DAT',TYPE='NEW')
         WRITE(1,*) (TEMP1(I),TEMP2(I),I=2,513)
C
C        get the present time
C
         CALL TIME(A)
C
C        type the present time on the terminal
C
         TYPE 1004,(A(I),I=1,4)
C
C        integrate the measured power spectrum
C
         POWER=1000.0*RINTGL(TEMP1,TEMP2,1024,512,I)
C
C        type the integral of the power spectrum
C
         TYPE 1008,POWER
1008     FORMAT(' ','The integral of the high frequency power spectrum'
     *   ,' is',2X,F8.3/'0',80('*'))
C
```

```
C       write the power to the plot file
C
        WRITE(1,*) POWER
        CLOSE(UNIT=1,DISP='KEEP')
C
C       write the integral of the power spectrum to the print file
C
        WRITE(2,1008) POWER
C
C       write the header for the low-frequency end of the spectrum
C       to the print file
C
        WRITE(2,2005)
2005    FORMAT('1',80('*'),/'0',
     *  'The low-frequency part of the spectrum'/'0')
C
C       compute the lower frequency end of the spectrum
C
C
C       zero the buffer that is used to accumulate the spectrum
C
        DO 112 I=1,1024
        TEMP2(I)=0.0
112     CONTINUE
C
C       calculate the number of data buffers to create
C
        K=ISIZE/64
C
C       test to see if the record is long enough to compute a
C       low-frequency spectrum
C
        IF(K.GE.1) GO TO 113
        TYPE 1009
1009    FORMAT(' ','The data record is to short to compute a'
     *  ,' low-frequency spectrum')
        GO TO 125
113     CONTINUE
        TYPE 1010,K
1010    FORMAT(' ','The number of new buffers created is',I5)
C
C       compute the cutoff frequency for the low-pass filter
C
        F=1.0/(128.0*DT)
C
C       calculate the filter constants and set the filter
C
        CALL FILTER(F,DT)
        TYPE 1011,F
1011    FORMAT(' ','The data will be filtered at ',F8.2,2X,'(hz)')
C
C       reset to read the file from the start
C
        CALL SLWCLS
        CALL SLWOPN
C
C       do the calculations
C
        DO 120 I=1,K
C
C       set up the new real part
C
        N=1
        DO 116 J=1,64
C
C       read in a buffer
```

```
C
      CALL SLWDMP(IMAG,32)
C
C     AC couple the signal
C
      CALL FASAC(IMAG,1024,IOFF)
C
C     filter the signal
C
      DO 114 L=1,1024
      CALL LPFILT(IMAG(L),IMAG(L))
114   CONTINUE
C
C     strip out the values for a lower sampling rate
C
      DO 115 L=1,1024,64
      IREAL(N)=IMAG(L)
      N=N+1
115   CONTINUE
116   CONTINUE
C
C     zero the imaginary part of the signal
C
      DO 117 J=1,1024
      IMAG(J)=0
117   CONTINUE
C
C     window the data record
C
      CALL HAM(IREAL,1024,1024)
C
C     compute the fast Fourier transform
C
      CALL FFT(IERROR,1024,IREAL,IMAG,0,ISCALE)
C
C     do error handling
C
      IF(IERROR.EQ.0) GO TO 118
C
C     if there is an error indication type a message on the terminal
C
      TYPE 1007,IERROR
118   CONTINUE
C
C     compute the power spectrum
C
      CALL POWRSP(1024,IREAL,IMAG,TEMP1)
C
C     compute the scaling factor
C
      FACT=2.0**(ISCALE*2)
C
C     accumulate the spectrum
C
      DO 119  J=1,1024
      TEMP2(J)=TEMP2(J)+FACT*TEMP1(J)
119   CONTINUE
120   CONTINUE
C
C     average the power spectrum for possitive frequencies
C
      N=1024
C
C     compute the relational scaling factor
C
C     Note: the relational scaling factor must be multiplied
```

```
C               by a factor of 64 to account for the frequency
C               shifting that has been done.
C
        FACT=(1.438202E-13/FLOAT(K))*64.0
        DO 121 I=2,512
        TEMP2(I)=(TEMP2(I)+TEMP2(N))
        TEMP2(I)=FACT*TEMP2(I)
        N=N-1
121     CONTINUE
        TEMP2(1)=2.0*FACT*TEMP2(1)
        TEMP2(513)=2.0*FACT*TEMP2(513)
C
C       fill TEMP1 with the frequencies that correspond to the power
C       spectrum in TEMP2
C
        FACT=1.0/(65536.0*DT)
C
C       set the initial value to zero
C
        TEMP1(1)=0.0
C
C       set the remaining values
C
        DO 122 I=2,513
        TEMP1(I)=FACT*FLOAT(I-1)
122     CONTINUE
C
C       smooth the low-frequency spectrum with a moving average
C
        DO 124 J=1,4
        TEMP2(1)=(TEMP2(1)+TEMP2(2)+TEMP2(3))/3.0
        TEMP2(2)=(TEMP2(1)+TEMP2(2)+TEMP2(3))/3.0
        DO 123 I=3,511
        TEMP2(I)=(TEMP2(I+1)+TEMP2(I)+TEMP2(I-1))/3.0
123     CONTINUE
        TEMP2(512)=(TEMP2(511)+TEMP2(512)+TEMP2(513))/3.0
        TEMP2(513)=(TEMP2(511)+TEMP2(512)+TEMP2(513))/3.0
124     CONTINUE
C
C
C       write out the power spectrum
C
        WRITE(2,2006)
2006    FORMAT('0',80('*')/'0',4(1X,'Frequency',2X,'Spectrum',1X)/' '
     *   ,4(2X,'(Hz)',6X,'(pow/Hz)',1X)/' ')
        WRITE(2,2003) (TEMP1(I),TEMP2(I),I=2,513)
        WRITE(2,2004) TEMP2(1)
C
C       write out the spectrum
C
        OPEN(UNIT=1,NAME='DL1:SNGPLT.DAT',TYPE='OLD',ACCESS='APPEND')
        WRITE(1,*) (TEMP1(I),TEMP2(I),I=2,513)
        CLOSE(UNIT=1,DISP='KEEP')
125     CONTINUE
C
C       get the present time
C
        CALL TIME(A)
C
C       type the present time on the terminal
C
        TYPE 1004,(A(I),I=1,4)
C
C       the power spectrum has now been computed
C
C-----------------------------------------------------------------------------
```

```
C
C       the probability distribution and probability density function
C       will be computed next
C
C------------------------------------------------------------------------
C
C       reset to read the file from the start
C
        CALL SLWCLS
        CALL SLWOPN
C
C       zero the array TEMP1
C
        DO 126 I=1,1024
        TEMP1(I)=0.0
126     CONTINUE
C
C       cycle through the data
C
        DO 127 I=1,ISIZE
C
C       read in a buffer of data
C
        CALL SLWDMP(IREAL,32)
C
C       do not AC couple the signal
C
C       preprocess the histogram
C
        CALL PHIST(IREAL,TEMP1)
C
C       the probability distribution has been preprocessed when this loop
C       is completed
C
127     CONTINUE
C
C       finish processing the probability distribution
C
        CALL PDENS(TEMP1)
C
C       compute the probability density function
C
        CALL PDF(TEMP1,TEMP2)
C
C       scale the probability density function
C
        FACT=STATS(2)/2.44E-3
        DO 128 I=1,1024
        TEMP2(I)=FACT*TEMP2(I)
128     CONTINUE
C
C       write out the data
C
        FACT=9.76E-3/STATS(2)
        MIN=-(2.44E-3*FLOAT(IOFF))/STATS(2)
        NEXT=MIN+FACT
        FACT=2.0*FACT
C
C       page the print file for the probability data
C
        WRITE(2,2007)
2007    FORMAT('1',80('*'))
C
C       write out the headers
C
        WRITE(2,2008)
```

```
2008    FORMAT('0','The computed probability distribution and probability',
     * ,' density function'/'0',2(7X,'U',7X,2(2X,'Probability',2X))/' ',
     * 2(6X,'(SD)',7X,'Distribution',5X,'Density',4X))
C
C       write out the data
C
        OPEN(UNIT=1,NAME='DL1:SNGPLT.DAT',TYPE='OLD',ACCESS='APPEND')
        DO 129 I=1,1024,2
        WRITE(2,2009) MIN,TEMP1(I),TEMP2(I),NEXT,TEMP1(I+1),TEMP2(I+1)
        WRITE(1,*) MIN,TEMP1(I),TEMP2(I),NEXT,TEMP1(I+1),TEMP2(I+1)
2009    FORMAT(' ',T1,2(5X,F6.3,7X,F9.6,6X,F9.6,3X))
        MIN=MIN+FACT
        NEXT=NEXT+FACT
129     CONTINUE
        CLOSE(UNIT=1,DISP='KEEP')
C
C       set the present time
C
        CALL TIME(A)
C
C       type the present time on the terminal
C
        TYPE 1004,(A(I),I=1,4)
C
C-----------------------------------------------------------------------------
C
C       this is the end of the program
C
C-----------------------------------------------------------------------------
C
C       close the files to exit cleanly
C
        CALL SLWCLS
        CLOSE(UNIT=2,DISP='PRINT')
        GO TO 131
130     CONTINUE
C
C       type an error message and halt
C
        TYPE 1012
1012    FORMAT(' The data file was not for a single channel',
     *  ', request halted')
C
C       close the open file
C
        CLOSE(UNIT=3,DISP='KEEP')
131     CONTINUE
        STOP
        END
```

D.7.2 The FASSUM subroutine

```
        .TITLE FASSUM
        .IDENT /FV01/
;
;**********************************************************************
;
;       THIS IS A MACRO OR FORTRAN CALLABLE SUBROUTINE. IT IS USED TO
;       CALCULATE THE SUM OF UP TO 5120. INTEGER WORDS AT VERY HIGH SPEED.
;
;
;       CALLING CONVENTIONS:
;
;
;       FORTRAN         CALL FASSUM(BUF,N,AVE)
```

```
;
;       ARGUMENTS          BUF - THE BUFFER OF DATA (INTEGER)
;                          N   - THE NUMBER OF 32. WORD BLOCKS IN BUFF
;                          AVE - THE TWO WORD INTEGE ARRAY FOR THE CALCULATED SUM
;                                 WORD #1 - THE NUMBER OF 32767. OVERFLOWS
;                                 WORD #2 - THE REMAINDER
;
;**********************************************************************
;
;       GLOBAL DATA DEFINITIONS
;
;**********************************************************************
;
        .GLOBL FASSUM                   ; ADDRESS OF THE FASAVE SUBROUTINE
        .GLOBL $SAVAL                   ; ADDRESS OF THE REGISTER SAVING SUBROUTINE
;
;**********************************************************************
;
;       THE START OF THE CODE
;
;**********************************************************************
;
FASSUM: JSR PC,$SAVAL                   ; SAVE ALL REGISTERS
        MOV (R5)+,R2                    ; GET THE NUMBER OF ARGUMENTS (UNNESSARY)
        MOV (R5)+,R2                    ; GET THE STARTING ADDRESS OF THE BUFFER
        MOV @(R5)+,R3                   ; GET THE NUMBER OF WORDS IN THE BUFFER
        MOV (R5)+,R4                    ; GET THE ADDRESS OF THE OUTPUT VALUE
        CLR R0                          ; CLEAR THE OVERFLOW ACCUMULATOR
        MOV #-32767.,R1                 ; CLEAR THE ACCUMULATOR
1$:     ADD (R2)+,R1                    ; SUM TO THE ACCUMULATOR
        BMI 2$                          ; IF NO OVERFLOW GO TO 2$
        ADD #1.,R0                      ; IF THERE WAS AN OVERFLOW ADD 1. TO R0
        SUB #32767.,R1                  ; READJUST R1
2$:     SOB R3,1$                       ; HAVE WE FINISHED SUMMING THE DATA
        TST R1                          ; IS R1 NEGATIVE
        BMI 3$                          ; IF IT IS BRANCH TO 3$
        ADD #1.,R0                      ; FIX R0 FOR THE LOST 32767.
        BR 4$                           ; EXIT SUBROUTINE
3$:     ADD #32767.,R1                  ; ADD 32767 TO R1 TO MAKE IT POSITIVE
4$:     MOV R0,(R4)+                    ; MOVE THE RESULTS TO THE OUTPUT LOCATIONS
        MOV R1,(R4)+
;
;       THIS IS THE END OF THE SUBROUTINE
;
        RTS PC
        .END
```

D.7.3 The FASAC subroutine

```
        .TITLE FASAC
        .IDENT /FV01/
;
;------------------------------------------------------------------------
;
;       This subroutine is FORTRAN or MACRO callable.  It is used to AC
;       couple the data buffers that are collected by the LPA11-K if the
;       offset is known.
;
;       Calling conventions
;
;       FORTRAN         CALL FASAC(BUF,N,IOFF)
;
;       arguments       BUF  - an integer buffer containing the biased input
```

```
;                                      ;returned with bias removed
;                                N     - the integer size of BUF
;                                IOFF  - the integer value of the bias
;
;-----------------------------------------------------------------------------
;
;       global definition section
;
;-----------------------------------------------------------------------------
;
        .GLOBL  FASAC       ; address of the FASAC subroutine
        .GLOBL  $SAVAL      ; address of the register saving subroutine
;
;-----------------------------------------------------------------------------
;
;       start of FASAC code
;
;-----------------------------------------------------------------------------
;
FASAC:  JSR PC,$SAVAL       ; save all registers
        MOV (R5)+,R0        ; get the number of arguments (unnessesary)
        MOV (R5)+,R1        ; get the starting address of the data buffer
        MOV @(R5)+,R2       ; get the size of the buffer
        MOV @(R5)+,R3       ; get the offset
1$:     SUB R3,(R1)+        ; subtract the offset from the data word
        SOB R2,1$           ; are we finished
        RTS PC              ; return
;
;-----------------------------------------------------------------------------
;
;       end of code
;
;-----------------------------------------------------------------------------
;
        .END
```

D.7.4 The FASSTS subroutine

```
        .TITLE FASSTS
        .IDENT /FV01/
;
;-----------------------------------------------------------------------------
;
;       This is a FORTRAN or MACRO callable subroutine.  FASSTS is used
;       to quickly calculate the first four moments of a data buffer.
;
;       Calling conventions:
;
;       FORTRAN         CALL FASSTS(BUF,N,OUTPUT)
;
;       Arguments       BUF    - the integer data buffer
;                       N      - the integer size of buf
;                       OUTPUT - an output buffer of 6. double precision
;                                floating point values
;                                        #1 - mean first moment
;                                        #2 - mean second moment
;                                        #3 - mean third moment
;                                        #4 - mean fourth moment
;                                        #5 - mean fifth moment
;                                        #6 - mean sixth moment
;-----------------------------------------------------------------------------
;
```

```
        AC0=%0          ;
        AC1=%1          ;
        AC2=%2          ; this section defines the floating point
        AC3=%3          ; accumulators
        AC4=%4          ;
        AC5=%5          ;
;
;------------------------------------------------------------------------
;
;       global definitions
;
;------------------------------------------------------------------------
;
        .GLOBL FASSTS   ; address of the FASSTS subroutine
        .GLOBL $SAVAL   ; address of the register saving routine
;
;------------------------------------------------------------------------
;
;       start of FASSTS code
;
;------------------------------------------------------------------------
;
FASSTS: JSR PC,$SAVAL   ; save all registers
        STFPS -(SP)     ; save the floating point status
        MOV (R5)+,R0    ; get the number of arguments (unnessesary)
        MOV (R5)+,R0    ; get the starting address of the data buffer
        MOV @(R5)+,R1   ; get the size of the data buffer
        MOV R1,R3       ; save the size of the data buffer in R3
        MOV (R5)+,R2    ; get the starting address of the output buffer
        SETD            ; set floating double mode
        CLRD AC0        ; clear AC0
        STD AC0,MEAN    ; clear storage location for the first moment
        STD AC0,SQR     ; clear storage location for the second moment
        STD AC0,CUBE    ; clear storage location for the third moment
        STD AC0,FOUR    ; clear storage location for the fourth moment
        STD AC0,FIVE    ; clear storage location for the fifth moment
        STD AC0,SIX     ; clear storage location for the sixth moment
1$:     LDCID (R0)+,AC0 ; load and conver a value from single word
                        ;    integer to double precision floating in AC0
        STD AC0,FLUC    ; store this value in location FLUC
        LDD MEAN,AC1    ; load the previous MEAN into AC1

        ADDD AC0,AC1    ; add the value of FLUC to AC1
        STD AC1,MEAN    ; store the new MEAN in MEAN
        MULD FLUC,AC0   ; get the square of FLUC
        LDD SQR,AC1     ; load the previous SQR into AC1
        ADDD AC0,AC1    ; add the square of FLUC to AC1
        STD AC1,SQR     ; store the new SQR in SQR
        MULD FLUC,AC0   ; get the cube of FLUC
        LDD CUBE,AC1    ; load the previous CUBE into AC1
        ADDD AC0,AC1    ; add the cube of FLUC to AC1
```

```
        STD AC1,CUBE    ; store the new CUBE in CUBE
        MULD FLUC,AC0   ; get the 4-th power of FLUC
        LDD FOUR,AC1    ; load the previous FOUR into AC1
        ADDD AC0,AC1    ; add the 4-th power of FLUC to AC1
        STD AC1,FOUR    ; store the new FOUR in FOUR
        MULD FLUC,AC0   ; get the 5-th power of FLUC
        LDD FIVE,AC1    ; load the previous  FIVE into AC1
        ADDD AC0,AC1    ; add the  5-th power of FLUC to AC1
        STD AC1,FIVE    ; store the new FIVE in FIVE
        MULD FLUC,AC0   ; get the 6-th power of FLUC
        LDD SIX,AC1     ; load the previous SIX in AC1
        ADDD AC0,AC1    ; add the 6-th power of FLUC to AC1
        STD AC1,SIX     ; store the new SIX in SIX
        SOB R1,1$       ; have we finished
        LDCID R3,AC0    ; load and convert the size of BUF into AC0
        LDD MEAN,AC1    ; recall MEAN
        DIVD AC0,AC1    ; normalize MEAN
        STD AC1,(R2)+   ; return the first moment
        LDD SQR,AC1     ; recall SQR
        DIVD AC0,AC1    ; normalize SQR
        STD AC1,(R2)+   ; return the second moment
        LDD CUBE,AC1    ; recall CUBE
        DIVD AC0,AC1    ; normalize CUBE
        STD AC1,(R2)+   ; return the third moment
        LDD FOUR,AC1    ; recall FOUR
        DIVD AC0,AC1    ; normalize FOUR
        STD AC1,(R2)+   ; return the fourth moment
        LDD FIVE,AC1    ; recall FIVE
        DIVD AC0,AC1    ; normalize FIVE
        STD AC1,(R2)+   ; return the fifth moment
        LDD SIX,AC1     ; recall SIX
        DIVD AC0,AC1    ; normalize SIX
        STD AC1,(R2)+   ; return the sixth moment
        LDFPS (SP)+     ; restore the floating point status
;
;-----------------------------------------------------------------------
;
;       this is the end of the code
;
;-----------------------------------------------------------------------
;
        RTS PC
;
;-----------------------------------------------------------------------
;
;       temporary data storage
;
;-----------------------------------------------------------------------
;
FLUC:   .FLT4 0.0       ; value of the fluctuation
MEAN:   .FLT4 0.0       ; mean value accumulator
SQR:    .FLT4 0.0       ; mean square accumulator
CUBE:   .FLT4 0.0       ; mean cube accumulator
FOUR:   .FLT4 0.0       ; mean fourth accumulator
FIVE:   .FLT4 0.0       ; mean fifth accumulator
SIX:    .FLT4 0.0       ; mean sixth accumulator
        .END
```

D.7.5 The STSCOR subroutine

```
C-----------------------------------------------------------------------
C
C       STSCOR is a subroutine that corrects the stats array given by
C       the subroutine FASSTS for offset error.
C
C       Calling convention:     CALL STSCOR(STATS)
C
C       Arguments:              STATS - the REAL*8 6 element array
C                                       that was obtained from the
C                                       subroutine FASSTS.
C                                         #1 - the mean
C                                         #2 - the mean square
C                                         #3 - the 3-rd moment
C                                         #4 - the 4-th moment
C                                         #5 - the 5-th moment
C                                         #6 - the 6-th moment
C
C-----------------------------------------------------------------------
C
        SUBROUTINE STSCOR(STATS)
        REAL*8 STATS(6)
        STATS(2)=STATS(2)-STATS(1)**2
        STATS(3)=STATS(3)-3.0*STATS(1)*STATS(2)-STATS(1)**3
        STATS(4)=STATS(4)-3.0*(STATS(1)**2)*STATS(2)
        STATS(4)=STATS(4)-4.0*STATS(1)*STATS(3)-STATS(1)**4
        STATS(5)=STATS(5)-STATS(1)**5
        STATS(5)=STATS(5)-7.0*(STATS(1)**3)*STATS(2)
        STATS(5)=STATS(5)-7.0*(STATS(1)**4)*STATS(3)
        STATS(5)=STATS(5)-5.0*STATS(1)*STATS(4)
        STATS(6)=STATS(6)-STATS(1)**6
        STATS(6)=STATS(6)-12.0*(STATS(1)**4)*STATS(2)
        STATS(6)=STATS(6)-14.0*(STATS(1)**3)*STATS(3)
        STATS(6)=STATS(6)-12.0*(STATS(1)**2)*STATS(4)
        STATS(6)=STATS(6)-6.0*STATS(1)*STATS(5)
        STATS(1)=STATS(1)*2.44E-3
        STATS(2)=SQRT(STATS(2))
        STATS(3)=STATS(3)/STATS(2)**3
        STATS(4)=STATS(4)/STATS(2)**4
        STATS(5)=STATS(5)/STATS(2)**5
        STATS(6)=STATS(6)/STATS(2)**6
        STATS(2)=STATS(2)*2.44E-3
        RETURN
        END
```

D.7.6 The FILT subroutine

```
C-----------------------------------------------------------------------
C
C       This subroutine is used to set up the filter constants for the
C       low-pass filter subroutine LPFILT.  It is called before LPFILT
C       can be called. Note that the subroutine SETFLT is called from
C       this routine and therefor must be supplied to the task builder.
C
C       Call convention         CALL FILT(F,T)
```

```
C
C       Arguments                       F  - the desired cut-off frequency in Hz
C                                       T  - the sampling period
C
C------------------------------------------------------------------------------C
        SUBROUTINE FILTER(F,T)
        REAL*8 F,T,PI,RTWO,W,AAA,BBB,CCC
        RTWO=1.414213562
        PI=3.141592654
        W=SIN(PI*F*T)/COS(PI*F*T)
        AAA=1.0+RTWO/W+1.0/(W**2)
        BBB=2.0*(1.0-1.0/(W**2))
        CCC=1.0-RTWO/W+1.0/(W**2)
        CALL SETFLT(AAA,BBB,CCC)
        RETURN
        END
```

D.7.7 The SETFILT subroutine

```
        .TITLE SETFLT
        .IDENT /FV01/
;
;------------------------------------------------------------------------------
;
;       SUBROUTINE SETFLT LOADS THE LOW-PASS FILTER CONSTANTS CALCULATED
;       IN FILTER FOR USE IN LPFILT.
;
;       CALLING CONVENTIONS
;
;       FORTRAN         CALL SETFILT(A,B,C)
;
;       ARGUMENTS       A,B,C   - A REAL*8 CONSTANTS USED BY LPFILT
;------------------------------------------------------------------------------
;
;       GLOBAL DEFINITION REGION
;
;------------------------------------------------------------------------------
;
        .GLOBL SETFLT    ; ADDRESS OF THE SETFLT ROUTINE
        .GLOBL $SAVAL    ; ADDRESS OF THE REGISTER SAVING SUBROUTINE
        .GLOBL AAA       ; ADDRESS OF THE LPFILT CONSTANT AAA
        .GLOBL BBB       ; ADDRESS OF THE LPFILT CONSTANT BBB
        .GLOBL CCC       ; ADDRESS OF THE LPFILT CONSTANT CCC
;
;------------------------------------------------------------------------------
;
;       START OF SETFLT CODE
;
;------------------------------------------------------------------------------
;
SETFLT: JSR PC,$SAVAL    ; SAVE ALL REGISTERS
        MOV (R5)+,R0     ; GET THE NUMBER OF ARGUMENTS
        MOV (R5)+,R1     ; GET THE STARTING ADDRESS OF A
        MOV (R1)+,AAA    ;
        MOV (R1)+,AAA+2  ; SET THE VALUE OF AAA IN LPFILT
        MOV (R1)+,AAA+4  ; TO THE VALUE OF A
        MOV (R1)+,AAA+6  ;
        MOV (R5)+,R1     ; GET THE STARTING ADDRESS OF B
        MOV (R1)+,BBB    ;
        MOV (R1)+,BBB+2  ; SET THE VALUE OF BBB IN LPFILT
        MOV (R1)+,BBB+4  ; TO THE VALUE OF B
        MOV (R1)+,BBB+6  ;
        MOV (R5)+,R1     ; GET THE STARTING ADDRESS OF C
        MOV (R1)+,CCC    ;
        MOV (R1)+,CCC+2  ; SET THE VALUE OF CCC IN LPFILT
        MOV (R1)+,CCC+4  ; TO THE VALUE OF C
        MOV (R1)+,CCC+6  ;
        RTS PC           ; RETURN FROM THE SUBROUTINE
```

```
;
;------------------------------------------------------------------------
;
;       END OF THE CODE
;
;------------------------------------------------------------------------
;
        .END
```

D.7.8 The LPFILT subroutine

```
        .TITLE LPFILT
        .IDENT /FV01/
;
;------------------------------------------------------------------------
;
;       THIS IS A FORTRAN OR MACRO CALLABLE SUBROUTINE.
;       LPFILT IS A LOW-PASS SECOND ORDER BUTTERWORTH FILTER
;       USING A IIR IMPLIMENTATION.
;
;       CALLING CONVENTIONS;
;
;       FORTRAN         CALL LPFILT(X,Y)
;
;       ARGUMENTS       X - THE CURRENT INPUT VALUE (INTEGER)
;                       Y - THE CURRENT OUTPUT VALUE (INTEGER)
;
;       NOTES;  1) LPFILT CAN ONLY BE CALLED AFTER THE SUBROUTINE
;                  FILTER HAS BEEN CALLED AND THE VALUES OF THE
;                  CONSTANTS HAVE BEEN ESTABLISHED AND STORED.
;               2) LPFILT HAS BEEN WRITTEN FOR THE EXPRESS PURPOSE
;                  OF FILTERING THE DATA STREAM FROM THE LPA11-K.
;               3) THE ROLL-OFF RATE OF A SECOND ORDER BUTTERWORTH
;                  FILTER IS 12 DB/OCTAVE , HENCE LPFILT HAS A ROLL-OFF
;                  RATE OF 12 DB/OCTAVE.  IF THIS IS INADEQUATE EITHER
;                  FILTER THE DATA STREAM MORE THAN ONCE OR USE A
;                  DIFFERENT FILTER IMPLIMENTATION.
;               4) NEVER USE LPFILT TO FILTER TWO DATA STREAMS AT THE
;                  SAME TIME.  IF YOU WANT TO FILTER TWO STREAMS PASS
;                  THROUGH ONE STREAM COMPLETELY BEFORE FILTERING THE
;                  SECOND STREAM.  FILTERING A DATA STREAM TWICE IS
;                  THE SAME AS FILTERING TWO STREAMS.
;               5) THE OUTPUT STREAM WILL CARRY A TRANSIENT FROM THE
;                  START OF THE OUTPUT STREAM FOR ABOUT 4 OR 5 TIME
;                  CONSTANTS OF THE FILTER (TIME CONST.=1/2*PI*FC).
;
;------------------------------------------------------------------------
;
        AC0=%0          ;
        AC1=%1          ;
        AC2=%2          ; THIS SECTION DEFINES THE FLOATING POINT
        AC3=%3          ;       ACCUMULATORS
        AC4=%4          ;
        AC5=%5          ;
;
;------------------------------------------------------------------------
;
;       GLOBAL DEFINITIONS
;
;------------------------------------------------------------------------
;
```

```
        .GLOBL  LPFILT    ; ADDRESS OF THE LPFILT SUBROUTINE
        .GLOBL  $SAVAL    ; ADDRESS THE THE REGISTER SAVING SUBROUTINE
        .GLOBL  AAA       ; ADDRESS OF FILTER CONSTANT A
        .GLOBL  BBB       ; ADDRESS OF FILTER CONSTANT B
        .GLOBL  CCC       ; ADDRESS OF FILTER CONSTANT C
;
;----------------------------------------------------------------------
;
;       START OF LPFILT CODE
;
;----------------------------------------------------------------------
;
LPFILT: JSR PC,$SAVAL     ; SAVE ALL REGISTERS
        STFPS -(SP)       ; SAVE THE FLOATING POINT STATUS
        MOV (R5)+,R0      ; GET NUMBER OF ADDRESS OF ARGUMENTS (UNNESSARY)
        MOV (R5)+,R0      ; GET ADDRESS OF INPUT POINT
        MOV (R5),R1       ; GET ADDRESS OF OUTPUT POINT
        SETD              ; SET FLOATING DOUBLE MODE
        LDCID (R0),AC0    ; LOAD AND CONVERT THE INPUT POINT FROM SINGLE WORD
                          ;  INTEGER TO FLOATING POINT DOUBLE
        STD AC0,AC2       ; STORE AC0 IN AC2
        ADDD X1,AC0       ; ADD THE LAST INPUT VALUE TO THE PRESENT INPUT VALUE
        ADDD X1,AC0       ; ADD IT AGAIN
        ADDD X2,AC0       ; ADD THE 2ND LAST INPUT TO THE SUM IN AC0
        LDD BBB,AC1       ; LOAD THE CONSTANT B IN AC1
        MULD Y1,AC1       ; MULTIPLY B BY THE LAST OUTPUT VALUE
        SUBD AC1,AC0      ; SUBTRACT THIS FROM AC0
        LDD CCC,AC1       ; LOAD THE CONSTANT C IN AC1
        MULD Y2,AC1       ; MULTIPLY C BY THE 2ND LAST OUTPUT VALUE
        SUBD AC1,AC0      ; SUBTRACT THIS FROM AC0
        DIVD AAA,AC0      ; DIVIDE AC0 BY THE CONSTANT A
        STCDI  AC0,(R1)   ; STORE THE INTERGER VALUE OF THE OUTPUT AT THE
                          ;   OUTPUT ADDRESS
        MOV Y1,Y2         ;
        MOV Y1+2,Y2+2     ;
        MOV Y1+4,Y2+4     ; SHIFT THE LAST OUTPUT INTO THE 2ND LAST OUPUT
        MOV Y1+6,Y2+6     ;
        MOV X1,X2         ;
        MOV X1+2,X2+2     ;
        MOV X1+4,X2+4     ; SHIFT THE LAST INPUT INTO THE 2ND LAST INPUT
        MOV X1+6,X2+6     ;
        STD AC0,Y1        ; SHIFT THE PRESENT OUTPUT INTO THE LAST OUTPUT
        STD AC2,X1        ; SHIFT THE PRESENT INPUT INTO THE LAST INPUT
        LDFPS (SP)+       ; RESTORE THE FLOATING POINT STATUS
        RTS PC            ; RETURN FROM SUBROUTINE
;
;----------------------------------------------------------------------
;
;       LPFILT CODE HAS FINISHED
;
;       DATA STORAGE ALLOCATION FOLLOWS
;
;----------------------------------------------------------------------
;
X1:     .FLT4   0.0       ; STORAGE FOR THE LAST INPUT
X2:     .FLT4   0.0       ; STORAGE FOR THE 2ND LAST INPUT
Y1:     .FLT4   0.0       ; STORAGE FOR THE LAST OUTPUT
Y2:     .FLT4   0.0       ; STORAGE FOR THE 2ND LAST OUTPUT
AAA:    .FLT4   3502.0    ; STORAGE FOR FILTER CONSTANT A
BBB:    .FLT4   -7000.0   ; STORAGE FOR FILTER CONSTANT B
CCC:    .FLT4   3502.0    ; STORAGE FOR FILTER CONSTANT C
;
;----------------------------------------------------------------------
;
;       END OF MAC FILE
;
;----------------------------------------------------------------------
;
        .END
```

D.7.9 Hamming window subroutine

```
C
C-----------------------------------------------------------------------
C
C       Subroutine HAM does Hamming windowing on a data buffer before
C       FFT processing. The window is set up to have a DC gain of 1.0.
C
C       Calling conventions      CALL HAM(VEC,IN,N)
C
C       Arguments                VEC - a integer data array
C                                IN  - the integer allocated size of VEC
C                                N   - the number of samples in VEC
C
C-----------------------------------------------------------------------
C
        SUBROUTINE HAM(VEC,IN,N)
C
C       type declaration section
C
        REAL PI,TM,T
        INTEGER VEC(IN),IN,N,I
        PI=3.14592654
        T=FLOAT(N)
        TM=PI/FLOAT(N)
        DO 10 I=1,N
        VEC(I)=IFIX(FLOAT(VEC(I))*(1.0+0.8519*COS(T*TM)))
        T=T+2.0
 10     CONTINUE
        RETURN
        END
```

D.7.10 The probability density function subroutines

D.7.10.1 Subroutine PHIST

```
C-----------------------------------------------------------------------
C
C       The subroutine PHIST is used to 'histogram' buffers of data collected
C       by the LPA11-K.  The histogram is done by dividing the raw integer
C       data by 4 and adding 1 to compute the address of the output array.
C       Then 1.0 is added to the value of the output array at that address
C       for each occurance of that address.
C
C       Calling convention       CALL PHIST(IVAL,VEC)
C
C       Arguments                IVAL - a 1024 word integer data buffer
C                                VEC  - a 1024 word real array containing
C                                       the histogram
C
C       Note:   PHIST only adds to the output vector so multiple calls
C               will accumulate the histogram.  If PHIST requires that
C               VEC must be zeroed if it is not to be accumulated to.
C
C-----------------------------------------------------------------------
C
        SUBROUTINE PHIST(IVAL,VEC)
        INTEGER I,J,K,IVAL(1024)
        REAL VEC(1024)
        DO 10 I=1,1024
        J=IVAL(I)/4
        J=J+1
        VEC(J)=VEC(J)+1.0
 10     CONTINUE
        RETURN
        END
```

D.7.10.2 Subroutine PDENS

```
C
C-----------------------------------------------------------------------
C
C       Subroutine PDENS is used to convert the output vector from PHIST
C       to a probability distribution.  It only works on the output from
C       PDENS.
C
C       Calling convention      CALL PDENS(VEC)
C
C       Arguments               VEC - the 1024 real array that contains the
C                                     histogram on input and the probability
C                                     distribution on output
C
C-----------------------------------------------------------------------
C
        SUBROUTINE PDENS(VEC)
        INTEGER I,J,K
        REAL VEC(1024)
        J=1023
10      CONTINUE
        K=J+1
        DO 20 I=1,J
        VEC(K)=VEC(K)+VEC(I)
20      CONTINUE
        J=J-1
        IF(J.GT.0) GO TO 10
        DO 30 I=1,1024
        VEC(I)=VEC(I)/VEC(1024)
30      CONTINUE
        RETURN
        END
```

D.7.10.3 Subroutine PDF

```
C
C-----------------------------------------------------------------------
C
C       Subroutine PDF calculates the probability density function from
C       the probability distribution by differentiation.  The input array
C       is created by previous calls to PHIST and PDENS.
C
C       Calling convention      CALL PDF(VEC,PDF)
C
C       Arguments               VEC - the 1024 real input array containing
C                                     the probability distribution
C                               PDF - the 1024 real output array containing
C                                     the probability density function
C
C-----------------------------------------------------------------------
C
        SUBROUTINE PDF(VEC,PDF)
        INTEGER I,J
        REAL VEC(1024),PDF(1024)
        PDF(1)=(VEC(3)/20.0+VEC(2)/40.0)
        PDF(2)=(VEC(4)/20.0+(VEC(3)-VEC(1))/40.0)
        DO 10 I=3,1022
        PDF(I)=((VEC(I+2)-VEC(I-2))/20.0+(VEC(I+1)-VEC(I-1))/40.0)
```

```
10      CONTINUE
        PDF(1023)=((1.0-VEC(1021))/20.0+(VEC(1024)-VEC(1022))/40.0)
        PDF(1024)=((1.0-VEC(1022))/20.0+(1.0-VEC(1023))/40.0)
        RETURN
        END
```

D.7.11 The RINGLT subroutines

```
C-----------------------------------------------------------------------
C
C       The subroutine RINTGL is a real function that calculates the integral
C       of a randomly distributed set of data points.  Five data points are
C       used to least squares fit a quadratic function.  This quadratic is
C       then used to calculate the differential area between the midway points
C       between the centermost point and its closest neighbours.  These
C       sub-areas are summed to give the integeral.
C
C       Calling convention:     CALL RINTGL(X,Y,IN,N,ER)
C
C       Arguments:              X       - the x-coordinate array (REAL*2)
C                               Y       - the y-coordinate array (REAL*2)
C                               IN      - the dimension of the x and y arrays
C                               N       - the number of data pairs
C                               ERR     - an error parameter that indicates the
C                                         the number of reoccurances of any
C                                         x value
C
C
C-----------------------------------------------------------------------
C
        REAL FUNCTION RINTGL(X,Y,IN,N,ER)
        INTEGER IN,N,I,J,K,ER,IJ(5)
        REAL X(IN),Y(IN),RINTGL,T1,T2,A,B,D,E,F,XX,YY,ZZ
        RINTGL=0.0
10      CONTINUE
        K=0
        ER=0
        DO 40 I=2,N
        IF(X(I).GE.X(I-1)) GO TO 20
        T1=X(I)
        T2=Y(I)
        X(I)=X(I-1)
        Y(I)=Y(I-1)
        X(I-1)=T1
        Y(I-1)=T2
        K=K+1
        GO TO 30
20      CONTINUE
        IF(X(I).NE.X(I-1)) GO TO 30
        ER=ER+1
30      CONTINUE
40      CONTINUE
        IF(K.NE.0) GO TO 10
        DO 160 I=1,N
        IF(I.GT.3) GO TO 70
        IJ(1)=1
        IJ(2)=2
        IJ(3)=3
        IJ(4)=4
        IJ(5)=5
        GO TO 110
```

```
70      CONTINUE
        J=N-I
        IF(J.GT.3) GO TO 100
        IJ(1)=N-4
        IJ(2)=N-3
        IJ(3)=N-2
        IJ(4)=N-1
        IJ(5)=N
        GO TO 110
100     CONTINUE
        IJ(1)=I-2
        IJ(2)=I-1
        IJ(3)=I
        IJ(4)=I+1
        IJ(5)=I+2
110     CONTINUE
        A=0.0
        B=0.0
        C=0.0
        D=0.0
        E=0.0
        F=0.0
        XX=0.0
        YY=0.0
        ZZ=0.0
        DO 120 J=1,5
        T1=X(IJ(J))
        T2=Y(IJ(J))
        A=A+T1**4
        B=B+T1**3
        C=C+T1**2
        E=E+T1
        XX=XX+T2*T1*T1
        YY=YY+T2*T1
        ZZ=ZZ+T2
120     CONTINUE
        F=((ZZ*A-XX*C)*(A*C-B*B)-(YY*A-XX*B)*(E*A-C*B))
        F=F/((5.0*A-C*C)*(A*C-B*B)-(E*A-C*B)**2)
        D=((YY*A-XX*B)-(E*A-C*B)*F)/((C*A-B*B)
        E=(XX-C*F-D*B)/A
        IF(I.EQ.1.OR.I.EQ.N) GO TO 130
        B=(X(I-1)+X(I))*0.5
        A=(X(I+1)+X(I))*0.5
        GO TO 150
130     CONTINUE
        IF(I.NE.1) GO TO 140
        B=X(1)
        A=(X(2)+X(1))*0.5
        GO TO 150
140     CONTINUE
        B=(X(N-1)+X(N))*0.5
        A=X(N)
150     CONTINUE
        RINTGL=RINTGL+E*((A**3)-(B**3))/3.0
        RINTGL=RINTGL+D*((A**2)-(B**2))*0.5
        RINTGL=RINTGL+F*(A-B)
160     CONTINUE
        RETURN
        END
```

D.7.12 The spectra and PDF plotting program

This program was used to plot the spectra and probability functions calculated in the single channel program.

```
        INTEGER CODE,I,IX,IY,LTYPE,PEN
        REAL X(1024),Y(1024),Y1(1024),XMIN,XMAX,YMIN,YMAX,POWER
        LOGICAL XLBL(40),YLBL(40),XLOG,YLOG,SYM
        LOGICAL SPEC,CPDF,PDF,SLPDF
        OPEN(UNIT=1,TYPE='OLD',NAME='DL1:INFO.DAT')
        READ(1,*) I,ITAPE,IBAND
        IF(I.NE.1) GO TO 999
        READ(1,*) ICHN
        READ(1,*) DT
        READ(1,*) DCPART,ACPART
        READ(1,*) SIZE
        CLOSE(UNIT=1,DISP='KEEP')
        OPEN(UNIT=1,NAME='DL1:SNGPLT.DAT',TYPE='OLD')
        I=SIZE/64
        IF(I.GE.1) GO TO 20
        READ(1,*) (X(I),Y(I),I=1,512)
        READ(1,*) POWER
        POWER=(ACPART**2)/POWER
        DO 10 I=1,512
        Y(I)=POWER*Y(I)
10      CONTINUE
        ISPC=512
        GO TO 60
20      CONTINUE
        ISPC=1024
        READ(1,*) (X(I),Y(I),I=513,1024)
        READ(1,*) POWER
        READ(1,*) (X(I),Y(I),I=1,512)
        DO 40 I=513,1024
        X(I)=1000.0*X(I)
40      CONTINUE
        POWER=(ACPART**2)/POWER
        DO 50 I=1,1024
        Y(I)=POWER*Y(I)
50      CONTINUE
60      CONTINUE
        OPEN(UNIT=2,NAME='DL:SNGLBL.TXT',TYPE='OLD')
        READ(2,2000) IX
2000    FORMAT(I2)
        READ(2,2010) (XLBL(I),I=1,40)
2010    FORMAT(40A1)
        READ(2,2000) IY
        READ(2,2010) (YLBL(I),I=1,40)
        CALL RNGBEL(5)
        TYPE 5060
5060    FORMAT(' ','Do you want a logarithmic plot of the '
     *  ,'spectrum ',$)
        ACCEPT 5050,SPEC
        IF(.NOT.SPEC) GO TO 90
        CALL SORT(X,Y,1024,ISPC,XMIN,XMAX,YMIN,YMAX)
        CODE=4
        CALL RNGBEL(5)
        TYPE 5000
```

```
5000    FORMAT(' Load an 11X17 paper in the plotter and hit return'
     *  ,' when the plotter is ready')
        ACCEPT 5010 ,I
5010    FORMAT(' ',I3)
        CALL PLTSET(CODE)
        XMIN=FLOAT(IFIX(ALOG10(XMIN)-0.9))
        YMIN=FLOAT(IFIX(ALOG10(YMIN)-0.9))
        XMAX=FLOAT(IFIX(ALOG10(XMAX)+0.9))
        YMAX=FLOAT(IFIX(ALOG10(YMAX)+0.9))
        XLOG=.TRUE.
        YLOG=.TRUE.
        CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
        DO 150 I=1,1024
        IF(X(I).GT.0.0) GO TO 110
        X(I)=XMIN-10.0
        GO TO 120
110     CONTINUE
        X(I)=ALOG10(X(I))
120     CONTINUE
        IF(Y(I).GT.0.0) GO TO 130
        Y(I)=YMIN-10.0
        GO TO 140
130     CONTINUE
        Y(I)=ALOG10(Y(I))
140     CONTINUE
150     CONTINUE
        LTYPE=8
        PEN=1
        SYM='*'
        CALL GRAPH(X,Y,ISPC,1024,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
90      CONTINUE
        DO 160 I=1,1024,2
        READ(1,*) (X(I),Y(I),Y1(I),X(I+1),Y(I+1),Y1(I+1))
160     CONTINUE
        READ(2,2000) IX
        READ(2,2010) (XLBL(I),I=1,40)
        READ(2,2000) IY
        READ(2,2010) (YLBL(I),I=1,40)
        CLOSE(UNIT=2,DISP='KEEP')
        CLOSE(UNIT=1,DISP='KEEP')
        XMIN=X(1)
        XMAX=X(1024)
        YMIN=0.0
        YMAX=1.0
        CALL RNGBELL(5)
        TYPE 5040
5040    FORMAT(' ','Do you want a linear plot of the cumulative '
     *  ,'probability distribution ',$)
        ACCEPT 5050,CPDF
        CPDF=.NOT.CPDF
5050    FORMAT(L1)
        TYPE 5030
5030    FORMAT(' ','Do you want a linear plot of the'
     *   ,' probability density function ',$)
        ACCEPT 5050,PDF
        PDF=.NOT.PDF
        IF(CPDF.AND.PDF) GO TO 163
        CODE=1
        CALL RNGBEL(5)
        TYPE 5020
5020    FORMAT(' Load a 8.5X11 paper in the plotter and '
     *  ,'hit a return when the plotter is ready')
        ACCEPT 5010 ,I
        CALL PLTSET(CODE)
        XLOG=.FALSE.
        YLOG=.FALSE.
```

```
          CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
          IF(CPDF) GO TO 161
          LTYPE=8
          PEN=2
          CALL GRAPH(X,Y,1024,1024,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
 161      CONTINUE
          IF(PDF) GO TO 162
          LTYPE=8
          PEN=3
          CALL GRAPH(X,Y1,1024,1024,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
 162      CONTINUE
 163      CONTINUE
          CALL RNGBEL(5)
          TYPE 5080
5080      FORMAT(' ','Do you want a semi-logarithmic plot of the PDF ',$)
          ACCEPT 5050,SLPDF
          IF(.NOT.SLPDF) GO TO 200
          YMAX=0.0
          YMIN=-4.0
          DO 190 I=1,1024
          IF(Y1(I).GT.0.0) GO TO 170
          Y1(I)=YMIN-10.0
          GO TO 180
 170      CONTINUE
          Y1(I)=ALOG10(Y1(I))
 180      CONTINUE
 190      CONTINUE
          CODE=2
          CALL RNGBEL(5)
          TYPE 5020
          ACCEPT 5010
          CALL PLTSET(CODE)
          XLOG=.FALSE.
          YLOG=.TRUE.
          CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
          LTYPE=8
          PEN=1
          CALL GRAPH(X,Y1,1024,1024,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
 200      CONTINUE
 999      CONTINUE
          CALL STRPEN
          STOP
          END
```

D.7.13 The autocorrelation program

```
C--------------------------------------------------------------------------
C
C         This program processes the data from a single channel of the
C         LPA-11K.  It accepts sampling rates up to the maximum that the
C         A/D converter can process.  The data is assumed to have been written
C         on disk drive 01 by the program FAST, which creates all of the
C         nessesary files on this disk.
C
C         This program only computes the autocorrelation by the direct method.
C         It is quite slow, taking approximately 2 to 2.5 hours for 2300 buffers
C         of data (1 buffer = 1024 samples).  Points are computed for 56
C         quasi-logrithmically distributed delays from 0 to 5120 sampling
C         intervals.
C
C--------------------------------------------------------------------------
C
C         Data declaration and definition region
C
C--------------------------------------------------------------------------
C
          INTEGER BLK(10240),K(56),I,J,L,ISIZE,ITAPE,ICHN,IBAND
C
C         BLK    -  a 10240 word buffer used to store the signal
C         K      -  a 56 word buffer used to store the integer delays
C         I,J,L  -  dummy variables
```

```
C       ISIZE -  the number of data buffers, modified during execution
C                to 1/5 the initial number
C       ITAPE -  the tape number used to identify the data
C       ICHN  -  the channel number used to identify the data
C       IBAND -  the band number used to identify the data
C
        INTEGER A(4),B(5),IOFF,K1(2)
C
C       A      - a 4 word buffer used to store the ascii code for the time
C       B      - a 5 word buffer used to store the ascii code for the date
C       IOFF   - the integer value of the offset
C       K1     - a 2 word buffer used to compute the integral (sum) of the
C                signal
C
        REAL*8 R(56),TAU(56),TEMP,M
C
C       R      - a 56(X4) word buffer used to store the autocorrelation
C       TAU    - a 56(X4) word buffer used to store the time delays
C       TEMP   - a temperary storage location used to calculate the mean
C       M      - the mean value of the signal
C
        REAL DT,DCPART,ACPART,FCUT
C
C       DT     - the sampling interval
C       DCPART- the DC part of the intial signal (m/sec,etc)
C       ACPART- the AC part of the intial signal (m/sec,etc)
C       FCUT   - the cut-off frequency used for low-pass filtering
C
C-----------------------------------------------------------------------
C
C       Subroutine declarations
C
C-----------------------------------------------------------------------
C
        INTEGER SLWDMP,SLWOPN,SLWCLS
C
C       SLWOPN- the subroutine that opens the data file to disk 01
C       SLWDMP- the subroutine that reads the data file
C       SLWCLS- the subroutine that closes the data file
C
        INTEGER PTCOR,MEAN,BLKSFT,FASSUM,FASAC
C
C       PTCOR - the subroutine that calculates the correlation between two
C               buffers
C       MEAN  - the subroutine that calculates the mean of a buffer of values
C       BLKSFT- the subroutine that shifts a block of data from one location
C               location to another
C       FASSUM- the subroutine that integrates the data signal
C       FASAC - the subroutine that AC couples the signal
C
        INTEGER FILTER,LPFILT
C
C       FILTER- the subroutine that computes the filter constants and
C               sets the filter LPFILT
C       LPFILT- the subroutine that low-pass filters the signal,
C               the roll of rate is 12 db/oct
C
        INTEGER TIME,DATE,IFIX
C
C       TIME  - the subroutine that returns the ascii code for the time
C       DATE  - the subroutine that returns the ascii code for the date
C       IFIX  - the subroutine that converts reals to integers
C
        REAL FLOAT
C
C       FLOAT - the subroutine that converts integers to reals
C
        REAL*8 DINTGL
C
C       DINTGL- the subroutine that calculates the integral
C
```

```
      CALL TIME(A)
      CALL DATE(B)

      TYPE 5000,(A(I),I=1,4),(B(I),I=1,5)
5000  FORMAT(' ',80('*')/'0','Data processing commenced at '
     * ,4A2,' on ',5A2/' ')
      OPEN(UNIT=1,TYPE='OLD',NAME='DL1:INFO.DAT')
C
C     read in the the number of channels sampled,tape number and the
C     band number of the input signal
C
      READ(1,*) I,ITAPE,IBAND
C
C     if I is not equal to 1 go to 80 type a message and halt
C
      IF(I.NE.1) GO TO 80
C
C     read in the channel number
C
      READ(1,*) ICHN
C
C     read in the sampling interval
C
      READ(1,*) DT
C
C     read in the DC and AC parts of the input signal
C
      READ(1,*) DCPART,ACPART
C
C     read in the number of 1024 word buffers
C
      READ(1,*) ISIZE
C
C     close the information file
C
      CLOSE(UNIT=1,DISP='KEEP')
C
C     type the tape,channel and band nubers, and the number of 1024
C     word blocks of data on the terminal
C
      TYPE 5010,ITAPE,ICHN,IBAND,ISIZE
5010  FORMAT(' ','For tape ',I2,' channel ',I2,' band ',I2
     * /' ','With ',I4,' buffers (1024 words) of data')
C
C     type the sampling interval and the mean and RMS values of the
C     input signal on the terminal
C
      TYPE 5020,DT,DCPART,ACPART
5020  FORMAT(' ','The input data was sampled every',3X,E10.3,' sec'//' '
     * ,'The mean of the input signal was ',F8.3/' '
     * ,'The RMS of the input signal was  ',f8.3/'0')
C
C     reset ISIZE for use in the correlation analysis
C
C     Note: the analysis works on 5k buffers so ISIZE must be divided
C           by 5 and one 5k buffer cannot be included in the accumulation
C
      ISIZE=(ISIZE/5)-1
C
C     set the values of the integer delays
C
C     Note: the values are the integer value closest to a 1/3 octave
C           step from 5120 down
C
      K(1)=0
      K(2)=1
      K(3)=2
      K(4)=3
      K(5)=4
      K(6)=5
      K(7)=6
      K(8)=7
      K(9)=8
      K(10)=9
      K(11)=10
      K(12)=11
      K(13)=13
```

```
      K(14)=15
      K(15)=18
      K(16)=20
      K(17)=23
      K(18)=27
      K(19)=31
      K(20)=35
      K(21)=41
      K(22)=47
      K(23)=54
      K(24)=62
      K(25)=71
      K(26)=81
      K(27)=93
      K(28)=107
      K(29)=123
      K(30)=141
      K(31)=162
      K(32)=186
      K(33)=213
      K(34)=245
      K(35)=281
      K(36)=323
      K(37)=371
      K(38)=426
      K(39)=489
      K(40)=561
      K(41)=645
      K(42)=740
      K(43)=850
      K(44)=976
      K(45)=1128
      K(46)=1286
      K(47)=1477
      K(48)=1695
      K(49)=1947
      K(50)=2235
      K(51)=2566
      K(52)=2946
      K(53)=3383
      K(54)=3884
      K(55)=4459
      K(56)=5120
C
C     open the data file to compute the integer off-set
C
      CALL SLWOPN
C
C     intialize the mean value
C
      M=0.0
C
C     integrate the signal to get the off-set
C
      DO 10 I=1,ISIZE
C
C     fill  BLK with 160X32 words of data
C
      CALL SLWDMP(BLK,160)
C
C     calculate the intermediate sum
C
      CALL FASSUM(BLK,5120,K1)
C
C     accumulate the mean value
C
      M=M+32767.0*FLOAT(K1(1))+FLOAT(K1(2))
10    CONTINUE
C
C     calculate the mean input value
C
      M=M/(FLOAT(ISIZE)*5120.0)
C
C     compute the integer off-set
C
      IOFF=IFIX(M)
C
C     reset the data file to read from the beginning
C
      CALL SLWCLS
      CALL SLWOPN
C
C----------------------------------------------------------------------
```

```
C
C     compute the autocorrelation coefficients
C
C-------------------------------------------------------------------------
C
      CALL SLWOPN
C
C     zero the initial value of the correlation
C
      DO 20 I=1,56
      R(I)=0.0
20    CONTINUE
C
C     zero the intial value of the mean
C
      M=0.0
C
C     read in the first 5120 word buffer
C
      CALL SLWDMP(BLK,160)
C
C     AC couple the signal
C
      CALL FASAC(BLK,5120,IOFF)
C
C     start accumulating the correlations and mean
C
      DO 40 I=1,ISIZE
C
C     read in another 5120 buffer in order to do the accumulations
C
      CALL SLWDMP(BLK(5121),160)
C
C     AC couple the signal
C
      CALL FASAC(BLK(5121),5120,IOFF)
C
C     call PTCOR for each of the delays and accumulate them in R
C
      DO 30 J=1,56
      CALL PTCOR(BLK(1),BLK(K(J)+1),5120,TEMP)
      R(J)=R(J)+TEMP
30    CONTINUE
C
C     call MEAN to calculate the local mean and accumulate it in M
C
      CALL MEAN(BLK(1),5120,TEMP)
      M=M+TEMP
C
C     call BLKSFT to shift the second buffer into the possition of the
C     first buffer
C
      CALL BLKSFT(BLK(5121),BLK(1),5120)
C
C     this is the end of the loop that accumulates the correlations
C
40    CONTINUE
C
C     close the data file
C
      CALL SLWCLS
C
C     set the integer value of the number of accumulations in TEMP
C
      TEMP=FLOAT(ISIZE)
C
C     calculate the mean value of the digital signal
C
      M=M/TEMP
C
C     calculate the value of the correlation due to the mean
C
      M=M*M
```

```
C
C       calculate the average correlation  minus the value due to the
C       mean value the signal
C
        DO 50 I=1,56
        R(I)=(R(I)/TEMP)-M
50      CONTINUE
C
C       calulate the scaling factor used to calculate the autocorrelation
C       coefficient
C
        TEMP=1.0/R(1)
C
C       set the zero delay correlation coefficient to 1.0
C
        R(1)=1.0
C
C       divide the time delayed correlations by the zero delay correlation
C
        DO 60 I=2,56
        R(I)=R(I)*TEMP
60      CONTINUE
C
C       convert the sampling interval to mille seconds
C
        DT=1000.0*DT
C
C       calculate the delays and store them in TAU
C
        DO 70 I=1,56
        TAU(I)=DT*FLOAT(K(I))
70      CONTINUE
C
C       calculate the integral time delay
C
        M=DINTGL(TAU,R,56,56,I)
C
C       open the print file on logical unit 2
C
        OPEN(UNIT=2,NAME='PRINT.TXT',TYPE='NEW')
C
C       start a new page
C
        WRITE(2,2000)
2000    FORMAT('1')
C
C       write the present time and date
C
        WRITE(2,5000) (A(I),I=1,4),(B(I),I=1,5)
C
C       write the tape,channel and band number and the number of 5120
C       sample buffers used to calculate the correlation coefficients
C
        WRITE(2,5010) ITAPE,ICHN,IBAND,ISIZE
C
C       convert the sampling interval back to seconds
C
        DT=DT/1000.0
C
C       write the sampling interval and the mean and RMS values to the
C       initial signal
C
        WRITE(2,5020) DT,DCPART,ACPART
C
C       write the integral time scale
C
        WRITE(2,2010) M
2010    FORMAT('0',80('*')/'0','The correlation results'/'0'
     *  ,'Integral time scale'2X,F9.4,2X,'(msec)'/'0',
     * 2(3X,'Time delay',4X,'Correlation Coefficient'1X)/' ',
     * 2(5X,'(msec)',27X)/' ')
C
```

```
C       write the time delays and the corresponding autocorrelation
C       coefficients
C
        OPEN(UNIT=1,NAME='DL1:CORREL.DAT',TYPE='NEW')
        WRITE(2,2020) (TAU(I),R(I),I=1,56)
        WRITE(1,*) (TAU(I),R(I),I=1,56)
2020    FORMAT(' ',I1,2(3X,F8.2,12X,F10.6,8X))
        CLOSE(UNIT=1,DISP='KEEP')
C
C       close the print file
C
        CLOSE(UNIT=2,DISP='PRINT')
        GO TO 90
80      CONTINUE
C
C       print an error message to indicate that the data file was not for
C       a single channel
C
        TYPE 5030
5030    FORMAT(' The data file was not for a single channel',
     *  ', request halted')
C
C       close unit 1
C
        CLOSE(UNIT=1,DISP='KEEP')
90      CONTINUE
C
C-----------------------------------------------------------------------------
C
C       this is the end of the program
C
C-----------------------------------------------------------------------------
C
        STOP
        END
```

D.7.14 The PTCOR subroutine

```
        .TITLE PTCOR
        .IDENT /FV01/
;-----------------------------------------------------------------------------
;
;       The subroutine PTCOR computes the correlation between to buffers.
;
;       Calling conventions:
;
;       FORTRAN         CALL PTCOR(VEC1,VEC2,N,COR)
;
;       Arguments:      VEC1 - the first vector (integer)
;                       VEC2 - the second vector (integer)
;                       N    - the minimum number of elements in either the
;                              first or second vectors (integer)
;                       COR  - the correlation (a real double precision number)
;
;-----------------------------------------------------------------------------
;
        AC0=%0          ; set AC0 as floating point accumulator 0
        AC1=%1          ; set AC1 as floating point accumulator 1
        AC2=%2          ; set AC2 as floating point accumulator 2
        .GLOBL PTCOR    ; the address of PTCOR
        .GLOBL $SAVAL   ; the address of the register saving subroutine
PTCOR:  JSR PC,$SAVAL   ; save all registers
        STFPS -(SP)     ; save the floating point status
```

```
        SETD            ; set for double precision
        MOV (R5)+,R0    ; get the number of arguments (unnessesary)
        MOV (R5)+,R0    ; get the address of the first vector
        MOV (R5)+,R1    ; get the address of the second vector
        MOV @(R5)+,R2   ; get the number of elements to accumulate over
        MOV (R5)+,R3    ; get the output address for the correlation
        MOV R2,R4       ; save the number of elements
        LDCID #0,AC0    ; zero floating point accumulator 0
1$:     LDCID (R0)+,AC1 ; load and convert an element of the first vector
                        ; into floating point accumulator 1
        LDCID (R1)+,AC2 ; load and convert an element of the second vector
                        ; into floating point accumulator 2
        MULD AC2,AC1    ; multiply AC1 by AC2 and store in AC1
        ADDD AC1,AC0    ; accumulate the product in AC0
        SOB R2,1$       ; decrement the number of elements and branch to
                        ; 1$ if there are some left
        LDCID R4,AC1    ; load and convert the number of elements into AC1
        DIVD AC1,AC0    ; divide AC0 by AC1 and store in AC0
        STD AC0,(R3)    ; store the correlation in the location pointed to
                        ; by R3
        LDFPS (SP)+     ; restore the floating point status
        RTS PC          ; return from the subroutine
        .END
```

D.7.15 The MEAN subroutine

```
        .TITLE MEAN
        .IDENT /FV01/
;
;-------------------------------------------------------------------------------
;
;       This subroutine calculates the mean value of a buffer.
;
;       Calling conventions:
;
;       FORTRAN         CALL MEAN(BUF,N,TEMP)
;
;       Arguments:      BUF  - the input buffer (integer)
;                       N    - the number of elements in the buffer (integer)
;                       TEMP - the mean value ( real double precision )
;
;-------------------------------------------------------------------------------
;
        AC0=%0          ; set AC0 to floating point accumulator 0
        AC1=%1          ; set AC1 to floating point accumulator 1
        .GLOBL MEAN     ; the address of subroutine mean
        .GLOBL $SAVAL   ; the address of the register saving subroutine
MEAN:   JSR PC,$SAVAL   ; save all registers
        STFPS -(SP)     ; save the floating point status
        SETD            ; set for floating double
        MOV (R5)+,R0    ; get the number of arguments (unnessesary)
        MOV (R5)+,R0    ; get the starting address of the input buffer
        MOV @(R5)+,R1   ; get the number of elements in the buffer
        MOV (R5)+,R2    ; get the address of the output value
        MOV R1,R3       ; save the number of elements
        LDCID #0,AC0    ; zero the accumulator
1$:     LDCID (R0)+,AC1 ; load and convert an element of the buffer into AC1
        ADDD AC1,AC0    ; add AC1 to AC0 and store it in AC0
        SOB R1,1$       ; decrement the number of elements and branch to 1$
                        ; if there are more
        LDCID R3,AC1    ; load and convert the number of elements into AC1
        DIVD AC1,AC0    ; divide AC0 by AC1 and store it in AC0
        STD AC0,(R2)    ; store the mean at the location pointed to by R2
        LDFPS (SP)+     ; restore the floating point status
        RTS PC          ; return from the subroutine
        .END
```

D.7.16 The BLKSFT subroutine

```
        .TITLE BLKSFT
        .IDENT /FV01/
;-------------------------------------------------------------------------------
;
;       Subroutine BLKSFT is a FORTRAN or  MACRO callable subroutine that
;       shifts large blocks of data from one buffer to another.
;
;       Calling conventions:
;
;       FORTRAN         CALL BLKSFT(SOURCE,DEST,N)
;
;       Arguments:      SOURCE - The source buffer (integer).
;                       DEST   - The destination buffer (integer).
;                       N      - The number of words to shift (integer).
;
;-------------------------------------------------------------------------------
;
        .GLOBL BLKSFT       ; the address of the subroutine BLKSFT
        .GLOBL $SAVAL       ; the address of the subroutine that saves the
                            ; registers
BLKSFT: JSR PC,$SAVAL       ; save all registers
        MOV (R5)+,R0        ; get the number of arguments (unnessesary)
        MOV (R5)+,R0        ; get the starting address of the source buffer
        MOV (R5)+,R1        ; get the starting address of the destination buffer
        MOV @(R5)+,R2       ; get the number of words to transfer
1$:     MOV (R0)+,(R1)+     ; move a word
        SOB R2,1$           ; decrement the number of words to transfer and
                            ; go to 1$ if there are more to transfer
        RTS PC              ; end the subroutine if there are no more words
                            ; to transfer
        .END
```

D.7.17 The autocorrelation plotting program

```
        INTEGER I,IX,IY,LTYPE,PEN,CODE
        REAL TAU(56),R(56),XMIN,XMAX,YMIN,YMAX
        LOGICAL XLOG,YLOG,XLBL(40),YLBL(40),SYM,LN,LG
        OPEN(UNIT=1,NAME='DL1:CORREL.DAT',TYPE='OLD')
        READ(1,*) (TAU(I),R(I),I=1,56)
        CLOSE(UNIT=1,DISP='KEEP')
        OPEN(UNIT=1,NAME='DL1:CORLBL.TXT',TYPE='OLD')
        READ(1,1000) IX
1000    FORMAT(I2)
        READ(1,1010) (XLBL(I),I=1,40)
1010    FORMAT(40A1)
        READ(1,1000) IY
        READ(1,1010) (YLBL(I),I=1,40)
        CLOSE(UNIT=1,DISP='KEEP')
        CALL SORT(TAU,R,56,56,XMIN,XMAX,YMIN,YMAX)
        YMAX=1.0
        YMIN=FLOAT(IFIX((YMIN-0.099)*10.0))/10.0
        XMAX=XMAX
        XMIN=XMIN
        CALL RNGBEL(5)
        TYPE 5030
5030    FORMAT(' Do you want a linear plot of the',
     *  ' auto correlation? ',$)
        ACCEPT 5040,LN
5040    FORMAT(L1)
        IF(.NOT.LN) GO TO 15
        CALL RNGBEL(5)
        TYPE 5000
```

```
5000     FORMAT(' ','Load a 8.5X11 paper in the plotter'/
      *  ' ','and hit a carrage return when it is ready',$)
         ACCEPT 5010
5010     FORMAT(' ')
         CODE=1
         CALL PLTSET(CODE)
         XLOG=.FALSE.
         YLOG=.FALSE.
         CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
         LTYPE=8
         PEN=1
         SYM='*'
         CALL GRAPH(TAU,R,56,56,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
15       CONTINUE
         CALL RNGBEL(5)
         TYPE 5050
5050     FORMAT(' Do you want a logarithmic plot of the',
      *  ' autocorrelation$ ',$)
         ACCEPT 5040,LG
         IF(.NOT.LG) GO TO 100
         DO 200 I=1,55
         TAU(I)=ALOG10(TAU(I+1))
200      CONTINUE
         XMIN=FLOAT(IFIX(TAU(1)-1.0))
         XMAX=FLOAT(IFIX(TAU(55)+1.0))
         XLOG=.TRUE.
         CALL RNGBEL(5)
         TYPE 5000
         ACCEPT 5010
         CODE=1
         CALL PLTSET(CODE)
         CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
         LTYPE=8
         PEN=1
         SYM='*'
         CALL GRAPH(TAU,R,55,56,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
100      CONTINUE
CALL STRPEN
STOP
END
```

D.7.18 The Zaric conditional sampling program

```
C------------------------------------------------------------------------
C
C        This program processes the data collected from a single channel of the
C        LPA-11K.  It accepts sampling rates up to the maximum that the A/D
C        converter can process.  The data is assumed to have been written on
C        disk drive 01 by the program FAST, which creates the all of the
C        nessesary files on this disk.  Zarics conditional sampling routine
C        is performed.
C
C------------------------------------------------------------------------
C
C        Data declaration and definition region
C
C------------------------------------------------------------------------
C
         LOGICAL XLOG,YLOG,XLBL(40),YLBL(40),SYM
C
C        XLOG     - is the X-axis logarithmic logical
C        YLOG     - is the Y-axis logarithmic logical
```

```
C       XLBL      - a 40 byte buffer for the X-axis lable
C       YLBL      - a 40 byte buffer for the Y-axis lable
C       SYM       - the symbol for the GRAPH subroutine
C
        INTEGER IREAL(1024),K1(2),IOFF
C
C       IREAL     - a 1024 word buffer used to store the real part of the signal
C                   or the spectrum
C       K1        - a 2 word buffer used to compute the integral (sum) of the
C                   signal
C       IOFF      - the integer value of the offset
C
        INTEGER ITAPE,ICHN,IBAND,ISIZE,I,J,K,L,N,ZONE
C
C       ITAPE     - the tape number, used to identify the data
C       ICHN      - the channel number, used to identify the data
C       IBAND     - the band number, used to identify the data
C       ISIZE     - the number of 1024 word buffers of data that were collected
C       I,J,K,L   - dummy variables
C       and, N
C       ZONE      - the intermitancy signal
C
        INTEGER A(4),B(5),LTYPE,PEN,CODE,IY,IX
C
C       A         - a 4 word buffer used to store the ascii code for the time
C       B         - a 5 word buffer used to store the ascii code for the date
C       LTYPE     - the integer code for the line type
C       PEN       - the pen number
C       CODE      - the integer paper code
C       IY        - the number of characters in YLBL
C       IX        - the number of characters in XLBL
C
        REAL PDF(1025),DT,DCPART,ACPART,RTCOND(8)
C
C       PDF       - a 1025(X2) word buffer used to store the spectrum,
C                   autocorrelation and probability distribution
C       DT        - the time interval between successive samples
C       DCPART    - the DC part of the initial signal (m/sec,etc)
C       ACPART    - the AC part of the initial signal (m/sec,etc)
C       RTCOND    - a 8(X2) word buffer used to set the gating parameters
C
        REAL TO,T1,T2,FACT,MIN,MEAN,NEXT,X1(256),Y1(256)
C
C       TO        - the initial time after midnight in seconds
C       T1,T2     - the elapsed time counters
C       FACT      - a temporary multiplying factor
C       MIN       - the minimum voltage considered for the probability
C                   distribution and probability density calculations,
C                   incremented for writing out these values
C       MEAN      - the mean input voltage
C       NEXT      - a temporary storage value
C
        REAL X(256),Y(256),XMIN,XMAX,YMIN,YMAX,MAX
C
C       X         - the X array for the graph subroutine
C       Y         - the Y array for the graph subroutine
C       XMIN      - the minimum X value
C       XMAX      - the maximum X value
C       YMIN      - the minimum Y value
C       YMAX      - the maximum Y value
C       MAX       - the maximum number of counts in the PDF
C
        REAL TEST(2),TCOND(5),RKUDU
C
C       TEST      - a 2 word buffer used to store the gating conditions
C       TCOND     - a 4 word buffer used to store the gating parameters
C       RKUDU     - the product of the derivative and u-u most probable
C
        REAL*8 STATS(6),H(6)
```

```
C
C       STATS    - a 6(x4) word buffer used to store the moments
C       H        - a 6(X4) word buffer used to accumulate the moments
C
C-------------------------------------------------------------------------
C
C       Subroutine declarations
C
C-------------------------------------------------------------------------
C
        INTEGER SLWOPN,SLWDMP,SLWCLS
C
C       SLWOPN   - the subroutine that opens the data file on disk 01
C       SLWDMP   - the subroutine that reads the data file
C       SLPCLS   - the subroutine that closes the data file
C
        INTEGER FASSUM,FASAC,FASSTS,STSCOR
C
C       FASSUM   - the subroutine that integrates the data signal
C       FASAC    - the subroutine that AC couples the signal
C       FASSTS   - the subroutine that computes the moments
C       STSCOR   - the subroutine that corrects the moments for residual offset
C
        INTEGER TSTSET,GATE,ZHIST,ZSET,DIFTOR,TDELAY
C
C       TSTSET   - the subroutine that sets the RICOND vector
C       GATE     - the subroutine that sets the intermitancy signal
C       ZHIST    - the subroutine that histograms the signal according to
C                  the intermitancy signal
C       ZSET     - the subroutine that sets ICOND from the RICOND vector
C       DIFTOR   - the subroutine that returns the derivative of the signal
C       TDELAY   - the subroutine that returns the time delayed signal
C
        INTEGER PLTSET,AXIS,GRAPH
C
C       PLTSET   - the subroutine that sets the plotter
C       AXIS     - the subroutine that draws the axis
C       GRAPH    - the subroutine that draws the graph
C
        INTEGER TIME,DATE,IFIX
C
C       TIME     - the subroutine that returns the ascii code for the time
C       DATE     - the subroutine that returns the ascii code for the date
C       IFIX     - the subroutine that converts reals to integers
C
        REAL SECNDS,FLOAT
C
C       SECNDS   - a subroutine that computes the time elapsed from some initial
C                  time
C       FLOAT    - the subroutine that converts integers to reals
C
C-------------------------------------------------------------------------
C
C       Start of the code
C-------------------------------------------------------------------------
C
C       set the intial time
C
        TO=SECNDS(0.0)
        CALL TIME(A)
        CALL DATE(B)
C
C       type the intial time and the present date to the terminal that ran the
C       program
C
        TYPE 1000,(A(I),I=1,4),(B(I),I=1,5)
1000    FORMAT(' ',80('*')/'0','Data processing commenced at ',4A2,' on '
     *  ,5A2/' ')
```

```
C
C-----------------------------------------------------------------------
C
C       this section gets the digitization and signal information that was
C       saved by the program FAST
C
C---------------------------------------------------------------------- -
C
C       open the information file that was written by FAST on disk DL0:
C
C       Note; logical unit number 3 must be assigned to DL1: when the
C             program is being Task Built. Use
C               ASN=DL1:3        in the options section of TKB
C
        OPEN(UNIT=3,TYPE='OLD',NAME='DL1:INFO.DAT')
C
C       read in the number of channels sampled,tape number and
C       the band number of the input signal
C
        READ(3,*) I,ITAPE,IBAND
C
C       if I is not equal to 1 type a message and halt
C
        IF(I.NE.1) GO TO 130
C
C       read in the tape channel number
C
        READ(3,*) ICHN
C
C       read in the sampling interval
C
        READ(3,*) DT
C
C       read in the DC and AC parts of the signal
C
        READ(3,*) DCPART,ACPART
C
C       read in the number of 1024 word blocks
C
        READ(3,*) ISIZE
C
C       close the information file
C
        CLOSE(UNIT=3)
C
C       type the tape, channel and band numbers,and the number
C       of 1024 word blocks of data on the terminal
C
        TYPE 1001,ITAPE,ICHN,IBAND,ISIZE
1001    FORMAT(' ','For tape ',I3,' channel ',I3,' band ',I3
     *   /' ','With ',I4,' buffers (1024 words) of data')
C
C       type the sampling interval and the mean and RMS values of the
C       signal on the terminal
C
        TYPE 1002,DT,DCPART,ACPART
1002    FORMAT(' ','The input data was sampled every',3X,E10.3,' sec'//' '
     * ,'The mean of the input signal was ',F8.3,1X,/' '
     * ,'The RMS of the input signal was  ',F8.3,1X,/'0')
C
C-----------------------------------------------------------------------
C
C       this section opens the print file named PRINT.TXT
C
C-----------------------------------------------------------------------
C
        OPEN(UNIT=2,NAME='PRINT.TXT',TYPE='NEW')
C
C       write in the lead information to the print file
C
```

```
      WRITE(2,2000)
2000  FORMAT('1')
      WRITE(2,1000) (A(I),I=1,4),(B(I),I=1,5)
      WRITE(2,1001) ITAPE,ICHN,IBAND,ISIZE
      WRITE(2,1002) DT,DCPART,ACPART
C
C-----------------------------------------------------------------------
C
C     this section initializes all of the nessesary arrays
C
C-----------------------------------------------------------------------
C
C     zero the gating parameter
C
      TCOND(1)=0.0
      TCOND(2)=0.0
      TCOND(3)=0.0
      TCOND(4)=0.0
C
C     zero the arrays used to set the gating parameters
C
      RTCOND(1)=0.0
      RTCOND(2)=0.0
      RTCOND(3)=0.0
      RTCOND(4)=0.0
      RTCOND(5)=0.0
      RTCOND(6)=0.0
      RTCOND(7)=0.0
      RTCOND(8)=0.0
C
C     intialize the intermitancy function to zero
C
      ZONE=0
C
C     set the X-axis lable
C
      IX=4
      XLBL(1)='u'
      XLBL(2)='/'
      XLBL(3)='u'
      XLBL(4)=''''
C
C     set the Y-axis lable
C
      IY=11
      YLBL(1)='P'
      YLBL(2)='r'
      YLBL(3)='o'
      YLBL(4)='b'
      YLBL(5)='a'
      YLBL(6)='b'
      YLBL(7)='i'
      YLBL(8)='l'
      YLBL(9)='i'
      YLBL(10)='t'
      YLBL(11)='y'
C
C-----------------------------------------------------------------------
C
C     this section computes the integer offset on the signal
C
C-----------------------------------------------------------------------
C
C     open the data file to compute the offset
C
      CALL SLWOPN
C
C     initialize the mean value
C
      MEAN=0.0
C
C     integrate the signal to get the offset
C
```

```
          DO 100 I=1,ISIZE
C
C         fill ireal with 32X32 words of data
C
          CALL SLWDMP(IREAL,32)
C
C         calculate the intermediate sum
C
          CALL FASSUM(IREAL,1024,K1)
C
C         accumulate the mean value
C
          MEAN=MEAN+32767.0*FLOAT(K1(1))+FLOAT(K1(2))
100       CONTINUE
C
C         calculate the mean input value
C
          MEAN=MEAN/(FLOAT(ISIZE)*1024.0)
C
C         compute the integer offset
C
          IOFF=IFIX(MEAN)
C
C         convert the mean input value to volts
C
          MEAN=(MEAN-2048.0)*2.44E-3
C
C         obtain the elapsed time
C
          T1=SECNDS(T0)
C
C         type the mean,offset and elapsed time on the terminal
C
          TYPE 1003,T1,MEAN,IOFF
1003      FORMAT(' ','Elapsed time',3X,F5.2,2X,'(sec)'/' ','Mean voltage'
     * ,2X,F6.3,2X,'(Volts)'/' ','Offset value',2X,I5/' ')
C
C         get the present time
C
          CALL TIME(A)
C
C         type the time onto the terminal
C
          TYPE 1004,(A(I),I=1,4)
1004      FORMAT(' ','The present time is ',4A2/' ')
C
C---------------------------------------------------------------------------------
C
C         this section computes the moments of the signal
C
C---------------------------------------------------------------------------------
C
C         get the number of seconds since midnight
C
          T1=SECNDS(0.0)
C
C         reset to read the file from the start
C
          CALL SLWCLS
          CALL SLWOPN
C
C         initialize the moments array
C
          DO 101 I=1,6
          STATS(I)=0.0
101       CONTINUE
C
C         accumulate over the whole data set
C
          DO 103 I=1,ISIZE
C
C         read in the buffer
C
          CALL SLWDMP(IREAL,32)
C
C         AC couple the signal
C
```

```
        CALL FASAC(IREAL,1024,IOFF)
C
C       compute the moments in the intermediate array H
C
        CALL FASSTS(IREAL,1024,H)
C
C       add the values in H to the values in STATS
C
        DO 102 J=1,6
        STATS(J)=STATS(J)+H(J)
102     CONTINUE
103     CONTINUE
C
C       average the moments
C
        DO 104 I=1,6
        STATS(I)=STATS(I)/FLOAT(ISIZE)
104     CONTINUE
C

C       correct the moments for offset error and non-dimensionalize the
C       third and higher moments
C
        CALL STSCOR(STATS)
C
C       get the elapsed time
C
        T2=SECNDS(T1)
C
C       convert the elapsed time to minutes
C
        T2=T2/60.0
C
C       type the elapsed time on the terminal
C
        TYPE 1005,T2
1005    FORMAT(' ','Elapsed time',T20,F7.3,2X,'(min)')
C
C       type the moments on to the terminal
C
        TYPE 1006,(STATS(I),I=2,6)
1006    FORMAT(' ','RMS',T20,F7.3,2X,'(Volts)'/' ','Skewness',T20,F7.3
     * /' ','Flatness',T20,F7.3/' ','Super-skewness',T20,F7.3/' ',
     * 'Hyper-skewness',T20,F7.3/' ')
C
C       write the moments out to the print file
C
        WRITE(2,1006) (STATS(I),I=2,6)
C
C       get the present time
C
        CALL TIME(A)
C
C       type the time onto the terminal
C
        TYPE 1004,(A(I),I=1,4)
C
C------------------------------------------------------------------------
C
C       start Zaric conditional sampling routine
C
C------------------------------------------------------------------------
C
C       find the most probable value
C
C
C       zero the histogram vector
C
        DO 190 I=1,1024
        PDF(I)=0.0
190     CONTINUE
C
C       open the data file
C
```

```
      CALL SLWOPN
C
C     loop through the data
C
      DO 200 I=1,ISIZE
C
C     read in a buffer of data
C
      CALL SLWDMP(IREAL,32)
C
C     histogram the data
C
      DO 210 J=1,1024
C
C     compute the histogram address
C
      K=((IREAL(J)-1)/4)+1
C
C     increment the histogram value
C
      PDF(K)=PDF(K)+1.0
210   CONTINUE
200   CONTINUE
C
C     close the data file
C
      CALL SLWCLS
C
C     search for the maximum
C
      MAX=0.0
      IMP=1
      DO 220 I=1,1024
      IF(PDF(I).LE.MAX) GO TO 230
      IMP=I
      MAX=PDF(I)
230   CONTINUE
220   CONTINUE
      IMP=(IMP*4)-2-IOFF
C
C     type the most probable value out on to the terminal
C
      TYPE 7000,IMP,MAX
7000  FORMAT(' ','The most probable point is at ',I4,' with ',
     *  ,F7.0,' counts'/' ')
C
C     write the most probable value to the print file
C
      WRITE(2,7000) IMP,MAX
C
C     zero the probability density function vector
C
      DO 250 I=1,1025
      PDF(I)=0.0
250   CONTINUE
C
C     open the data file
C
      CALL SLWOPN
C
C     read in a buffer
C
      CALL SLWDMP(IREAL,32)
C
C     AC couple the signal
C
      CALL FASAC(IREAL,1024,IOFF)
C
C     work through one buffer to eliminate the delay
C
      DO 320 I=1,1024
```

```
C
C         time delay the signal
C
          CALL TDELAY(IREAL(I),IVAL)
C
C         differentiate the signal
C
          CALL DIFTOR(IREAL(1),KUDU)
320       CONTINUE
C
C         calculate the test conditions
C
          DO 340 I=1,ISIZE-1
C
C         read in a buffer of data
C
          CALL SLWDMP(IREAL,32)
C
C         AC couple the signal
C
          CALL FASAC(IREAL,1024,IOFF)
C
C         work through that buffer
C
          DO 330 J=1,1024
C
C         time delay the signal
C
          CALL TDELAY(IREAL(J),IVAL)
C
C         differentiate the signal
C
          CALL DIFTOR(IREAL(J),KUDU)
C
C         calculate the gating signal
C
          RKUDU=FLOAT(KUDU)
C
C         separate and calculate the test conditions
          CALL ZSET(IVAL,IMP,RKUDU,RTCOND)
330       CONTINUE
340       CONTINUE
C
C         set the gating conditions
C
          CALL TSTSET(TCOND,RTCOND)
C
C         type out the RTCOND array to the terminal
C
          TYPE 7010,(I,RTCOND(1),I=1,8)
7010      FORMAT(' ','The RTCOND array contains '/' ',8('RTCOND(',
     *    I1,')',2X,E14.7/' '))
```

```
C
C       type out the TCOND array
C
        TYPE 7011,(I,TCOND(I),I=1,5)
7011    FORMAT(' ','The TCOND array contains '/' ',5('TCOND(',
     *  I1,')',2X,E14.7/' '))
C
C       write out the RTCOND array to the print file
C
        WRITE(2,7010) (I,RTCOND(I),I=1,8)
C
C       write out the TCOND array to the print file
C
        WRITE(2,7011) (I,TCOND(I),I=1,5)
C
C       reset IMP to its shifted value
C
        IMP=IMP+IOFF
C
C       close the data file
C
        CALL SLWCLS
C
C       open the data file
C
        CALL SLWOPN
C
C       work through a buffer
C
        CALL SLWDMP(IREAL,32)
        DO 350 I=1,1024
C
C       set the time delay
C
        CALL TDELAY(IREAL(I),IVAL)
C
C       set the derivative of the signal
C
        CALL DIFTOR(IREAL(I),KUDD)
350     CONTINUE
C
C       work through the rest of the data
C
        DO 370 I=1,ISIZE-1
C
C       read in a data buffer
C
        CALL SLWDMP(IREAL,32)
C
C       work through the data buffer
C
        DO 360 J=1,1024
C
C       set the time delayed signal
C
        CALL TDELAY(IREAL(J),IVAL)
C
C       set the derivative of the signal
C
        CALL DIFTOR(IREAL(J),KUDD)
C
C       set the second test parameter
C
        TEST(2)=FLOAT(IVAL-IMP)
C
C       set the first test parameter
C
        TEST(1)=TEST(2)*FLOAT(IABS(KUDD))
```

```
C
C       set the gating parameter
C
        CALL GATE(TEST,ZONE,ICOND)
C
C       histogram the data
C
        CALL ZHIST(IVAL,ZONE,PDF)
360     CONTINUE
370     CONTINUE
C
C       get the present time
C
        CALL TIME(A)
C
C       type the present time onto the terminal
C
        TYPE 1004,(A(I),I=1,4)
C

C       close the data file
C
        CALL SLWCLS
C
C       normalize the probability density functions
C
        MIN=RTCOND(1)/(16.0*PDF(1025))
        DO 380 I=1,1024
        PDF(I)=PDF(I)*MIN
380     CONTINUE
C
C       close the files to exit cleanly
C
        CALL SLWCLS
C
C-------------------------------------------------------------------------
C
C       plot the histograms
C
C-------------------------------------------------------------------------
C
        CALL RNGBEL(5)
        TYPE 5900
5900    FORMAT(' ','Put a 8.5X11 paper in the plotter' ,
     *  ,'and type a carriage return when it is ready'$)
        ACCEPT 5910
5910    FORMAT(' ')
        X(1)=-FLOAT(IOFF)/RTCOND(1)
        FACT=16.0/RTCOND(1)
        DO 390 I=2,256
        X(I)=X(I-1)+FACT
390     CONTINUE
        YMAX=1.0
        YMIN=0.0
        XMAX=6.0
        XMIN=-6.0
        CODE=1
        CALL PLTSET(CODE)
        K=1
        NEXT=0.0
        DO 400 I=1,256
        Y(I)=PDF(K)
        X1(I)=X(I)
        Y1(I)=Y(I)
        NEXT=NEXT+Y(I)
```

```
         K=K+1
400      CONTINUE
         NEXT=NEXT*(X(2)-X(1))
         XLOG=.FALSE.
         YLOG=.FALSE.
         CALL  AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,0,YLBL,LY)
         LTYPE=8
         PEN=2
         SYM='*'
         CALL  GRAPH(X,Y,256,256,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
         CALL  PDFMOM(X1,Y1,256,256,H)
         WRITE(2,7015)
7015     FORMAT('1',/' ','Zaric conditional sampling results.',/'0',
      *  ,'For the whole probability density function.')
         H(1)=H(1)/(2.44E-3*NEXT)
         H(2)=H(2)/(2.44E-3*SQRT(NEXT))
         WRITE(2,7040) (H(I),I=1,6),NEXT
         NEXT=0.0
         DO 410 I=1,256
         Y(I)=PDF(K)
         X1(I)=X(I)

         Y1(I)=Y(I)
         NEXT=NEXT+Y(I)
         K=K+1
410      CONTINUE
         NEXT=NEXT*(X(2)-X(1))
         PEN=3
         CALL  GRAPH(X,Y,256,256,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
         CALL  PDFMOM(X1,Y1,256,256,H)
         WRITE(2,7020)
7020     FORMAT('0','For the ejection events.',/' ')
         H(1)=H(1)/(NEXT*2.44E-3)
         H(2)=H(2)/(SQRT(NEXT)*2.44E-3)
         WRITE(2,7040) (H(I),I=1,6),NEXT
7040     FORMAT(' ','The mean',T30,F7.3/' ','The Rms',T30,F7.3/' ',
      *  'The skewness',T30,F7.3/' ','The flatness',T30,F7.3/' ',
      *  'The super-skewness',T30,F7.1/' ','The hyper-skewness',
      *  T30,F7.1/' ','Intermittancy',T30,F7.4)
         NEXT=0.0
         DO 420 I=1,256
         Y(I)=PDF(K)
         X1(I)=X(I)
         Y1(I)=Y(I)
         NEXT=NEXT+Y(I)
         K=K+1
420      CONTINUE
         NEXT=NEXT*(X(2)-X(1))
         PEN=4
         CALL  GRAPH(X,Y,256,256,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
         CALL  PDFMOM(X1,Y1,256,256,H)
         WRITE(2,7030)
7030     FORMAT('0','For the non-ejection non-sweep events.',/' ')
         H(1)=H(1)/(NEXT*2.44E-3)
         H(2)=H(2)/(SQRT(NEXT)*2.44E-3)
         WRITE(2,7040) (H(I),I=1,6),NEXT
         NEXT=0.0
         DO 430 I=1,256
         Y(I)=PDF(K)
         X1(I)=X(I)
         Y1(I)=Y(I)
         NEXT=NEXT+Y(I)
         K=K+1
```

```
430     CONTINUE
        NEXT=NEXT*(X(2)-X(1))
        PEN=5
        CALL GRAPH(X,Y,256,256,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
        CALL STRPEN
        CALL PDFMOM(X1,Y1,256,256,H)
        H(1)=H(1)/(NEXT*2.44E-3)
        H(2)=H(2)/(SQRT(NEXT)*2.44E-3)
        WRITE(2,7050)
7050    FORMAT('0','For the sweep events.'/' ')
        WRITE(2,7040) (H(I),I=1,6),NEXT
        CLOSE(UNIT=2,DISP='PRINT')
        GO TO 131
130     CONTINUE
C
C       type an error message and halt
C
        CALL RNGBEL(5)
        TYPE 1012
1012    FORMAT(' The data file was not for a single channel',
     *  ', request halted')
C
C       close the open file
C
        CLOSE(UNIT=3,DISP='KEEP')
131     CONTINUE
        STOP
        END
```

D.7.19 The ZHIST subroutine

```
        SUBROUTINE ZHIST(IVAL,ZONE,PDF)
        INTEGER IVAL,ZONE,I
        REAL PDF(1025)
        PDF(1025)=PDF(1025)+1
        I=((IVAL-1)/16)+1
        PDF(I)=PDF(I)+1.0
        I=I+512+ZONE*256
        PDF(I)=PDF(I)+1.0
        RETURN
        END
```

D.7.20 The GATE subroutine

```
        SUBROUTINE GATE(TEST,ZONE,TCOND)
        INTEGER ZONE
        REAL TCOND(5),TEST(2)
        IF(ZONE.NE.0) GO TO 30
        IF(TEST(1).GT.TCOND(1)) GO TO 10
        ZONE=-1
        GO TO 40
10      CONTINUE
        IF(TEST(1).LT.TCOND(2)) GO TO 20
        ZONE=1
20      CONTINUE
        GO TO 40
30      CONTINUE
        IF(ABS(TEST(2)).GT.TCOND(4)) GO TO 40
        IF(ABS(TEST(1)).GT.TCOND(4+ZONE)) GO TO 40
        ZONE=0
40      CONTINUE
        RETURN
        END
```

D.7.21 The TSTSET subroutine

```
      SUBROUTINE TSTSET(TCOND,RTCOND)
      REAL RTCOND(8),T,TCOND(5)
      T=RTCOND(7)+RTCOND(8)
      RTCOND(1)=SQRT(RTCOND(1)/T)
      RTCOND(4)=SQRT(RTCOND(4)/T)
      RTCOND(2)=SQRT(RTCOND(2)/RTCOND(7))
      RTCOND(3)=SQRT(RTCOND(3)/RTCOND(8))
      RTCOND(5)=SQRT(RTCOND(5)/RTCOND(7))
      RTCOND(6)=SQRT(RTCOND(6)/RTCOND(8))
      RTCOND(7)=RTCOND(7)/T
      RTCOND(8)=RTCOND(8)/T
      TCOND(1)=-RTCOND(5)
      TCOND(2)=RTCOND(6)
      TCOND(3)=0.3*RTCOND(5)
      TCOND(4)=2.0*RTCOND(1)
      TCOND(5)=0.3*RTCOND(6)
      RETURN
      END
```

D.7.22 The ZSET subroutine

```
      SUBROUTINE ZSET(IVAL,IMP,RKUDU,RTCOND)
      INTEGER IVAL,IMP
      REAL RTCOND(8),RIVAL,RKUDU
      RIVAL=FLOAT(IVAL)**2
      RKUDU=RKUDU**2
      RTCOND(1)=RTCOND(1)+RIVAL
      RTCOND(4)=RTCOND(4)+RKUDU
      RKUDU=(FLOAT(IVAL-IMP)**2)*RKUDU
      IF(IVAL.GT.IMP) GO TO 10
      RTCOND(2)=RTCOND(2)+RIVAL
      RTCOND(5)=RTCOND(5)+RKUDU
      RTCOND(7)=RTCOND(7)+1.0
      GO TO 20
10    CONTINUE
      RTCOND(3)=RTCOND(3)+RIVAL
      RTCOND(6)=RTCOND(6)+RKUDU
      RTCOND(8)=RTCOND(8)+1.0
20    CONTINUE
      RETURN
      END
```

D.7.23 The DIFTOR subroutine

```
        .TITLE DIFTOR
        .IDENT /X01/
;
;------------------------------------------------------------------------
;
;       This is a FORTRAN or MACRO callable subroutine.
;       DIFTOR is a broad band FIR differentiator given in 'One-Dimensional
;       Signal Processing' by Chi-Tsong Chen,Marcel Dekker Inc.,New York,
;       pg. 205.
;
;       Calling conventions:
;
;       FORTRAN              CALL DIFTOR(X,Y)
;
;       Arguments            X - the current input value (INTEGER)
;                            Y - the current output value (INTEGER)
;
```

```
;       Notes:  1) DIFTOR does not require a call to a setup subroutine
;                  and thus can be used as is.
;               2) DIFTOR has been written with the express purpose
;                  of differentiating the data stream from a LPA11-K.
;               3) Never use DIFTOR to differentiate two data streams
;                  the same time.  If you want to differentiate two
;                  streams pass through one stream completely before
;                  differentiating the second stream.  Differentiating
;                  twice is the same as differentiating two streams.
;               4) The output stream will contain a transient at the
;                  the begining.  Remember that DIFTOR has a FIR transfer
;                  function, thus the transient will settle out after the
;                  first 18 samples of the input stream.  The first 18
;                  samples of the output stream will be contaminated by
;                  input settling.
;
;-------------------------------------------------------------------------------
;
        AC0=%0          ;
        AC1=%1          ;
        AC2=%2          ; this section defines the floating point
        AC3=%3          ; accumulators
        AC4=%4          ;
        AC5=%5          ;
;
;------------------------------------------------------------------------------
;
;       global definitions
;
;------------------------------------------------------------------------------
;
        .GLOBL  DIFTOR  ; address of the DIFTOR subroutine
        .GLOBL  $SAVAL  ; address of the register saving subroutine
;
;-----------------------------------------------------------------------------
;
;       start of DIFTOR code
;
;----------------------------------------------------------------------------
;
DIFTOR: JSR PC,$SAVAL   ; save all registers
        STFPS -(SP)     ; save the floatin point resisters
        MOV (R5)+,R0    ; get the number of arguments (unnessary)
        MOV (R5)+,R0    ; get the address of the input point
        MOV (R5),R1     ; get the address of the output point
        SETD            ; set floating double
        LDCID (R0),AC0  ; load and convert the input point from single word
                        ; to floating double
        STD AC0,AC2     ; store AC0 in AC2
        MULD A,AC0      ; multiply the input point by constant A
        LDD X1,AC1      ; load the last input point into AC1
        STD AC2,X1      ; make the present point the next last point
        STD AC1,AC2     ; save the value in AC1
        MULD B,AC1      ; multiply the value in AC1 by constant B
        ADDD AC1,AC0    ; add this product to AC0
        LDD X2,AC1      ; load the 2-nd last point into AC1
        STD AC2,X2      ; make the last point the 2-nd last
        STD AC1,AC2     ; save the value in AC1
        MULD C,AC1      ; multiply the value in AC1 by constant C
        ADDD AC1,AC0    ; add this product to AC0
        LDD X3,AC1      ; load the 3-rd last point into AC1
        STD AC2,X3      ; make the 2-nd last point the 3-rd last
        STD AC1,AC2     ; save the value in AC1
        MULD D,AC1      ; multiply the value in AC1 by constant D
```

```
ADDD AC1,AC0      ; add this product to AC0
LDD X4,AC1        ; load the 4-th last point into AC1
STD AC2,X4        ; make the 3-rd last point the 4-th last
STD AC1,AC2       ; save the value in AC1
MULD E,AC1        ; multiply the value in AC1 by constant E
ADDD AC1,AC0      ; add this product to AC0
LDD X5,AC1        ; load the 5-th last point into AC1
STD AC2,X5        ; make the 4-th last point the 5-th last
STD AC1,AC2       ; save the value in AC1
MULD F,AC1        ; multiply the value in AC1 by constant F
ADDD AC1,AC0      ; add this product to AC0
LDD X6,AC1        ; load the 6-th last point into AC1
STD AC2,X6        ; make the 5-th last point the 6-th last
STD AC1,AC2       ; save the value in AC1
MULD G,AC1        ; multiply the value in AC1 by constant G
ADDD AC1,AC0      ; add this product to AC0
LDD X7,AC1        ; load the 7-th last point into AC1
STD AC2,X7        ; make the 6-th last point the 7-th last
STD AC1,AC2       ; store the value in AC1
MULD H,AC1        ; multiply the value in AC1 by constant H
ADDD AC1,AC0      ; add this product to AC0
LDD X8,AC1        ; load the 8-th last point into AC1
STD AC2,X8        ; make the 7-th last point the 8-th last
STD AC1,AC2       ; save the value in AC1
MULD I,AC1        ; multiply the value in AC1 by constant I
ADDD AC1,AC0      ; add this product to AC0
LDD X9,AC1        ; load the 9-th last point into AC1
STD AC2,X9        ; make the 8-th last point the 9-th last point
STD AC1,AC2       ; save the value in AC1
                  ; the 9-th last point is multiplied by 0.0
                  ; so we only save the value
LDD X10,AC1       ; load the 10-th last point into AC1
STD AC2,X10       ; make the 9-th last point the 10-th last
STD AC1,AC2       ; save the value in AC1
MULD J,AC1        ; multiply the value in AC1 by constant J
ADDD AC1,AC0      ; add this product to AC0
LDD X11,AC1       ; load the 11-th last point into AC1
STD AC2,X11       ; make the 10-th last point the 11-th last point
STD AC1,AC2       ; save the value in AC1
MULD K,AC1        ; multiply the value in AC1 by constant K
ADDD AC1,AC0      ; add this product to AC0
LDD X12,AC1       ; load the 12-th last point into AC1
STD AC2,X12       ; make the 11-th last point the 12-th last
STD AC1,AC2       ; save the value in AC1
MULD L,AC1        ; multiply the value in AC1 by constant L
ADDD AC1,AC0      ; add this product to AC0
LDD X13,AC1       ; load the 13-th last point into AC1
STD AC2,X13       ; make the 12-th last point the 13-th last
STD AC1,AC2       ; save the value in AC1
MULD M,AC1        ; multiply the value in AC1 by constant M
ADDD AC1,AC0      ; add this product to AC0
LDD X14,AC1       ; load the 14-th last point into AC1
STD AC2,X14       ; make the 13-th last point the 14-th last point
STD AC1,AC2       ; save the value in AC1
MULD N,AC1        ; multiply the value in AC1 by constant N
ADDD AC1,AC0      ; add this product to AC0
LDD X15,AC1       ; load the 15-th last point into AC1
STD AC2,X15       ; make the 14-th last point the 15-th last point
STD AC1,AC2       ; save the value in AC1
MULD O,AC1        ; multiply the value in AC1 by constant O
ADDD AC1,AC0      ; add this product to AC0
LDD X16,AC1       ; load the 16-th last point into AC1
STD AC2,X16       ; make the 15-th last point the 16-th last
STD AC1,AC2       ; save the value in AC1
MULD P,AC1        ; multiply the value in AC1 by constant P
ADDD AC1,AC0      ; add this product to AC0
LDD X17,AC1       ; load the 17-th last point into AC1
STD AC2,X17       ; make the 16-th last point the 17-th last
```

```
        STD AC1,AC2         ; save the value in AC1
        MULD Q,AC1          ; multiply the value in AC1 by constant Q
        ADDD AC1,AC0        ; add this product to AC0
        LDD X18,AC1         ; load the 18-th last point into AC1
        STD AC2,X18         ; make the 17-th last point the 18-th last
        MULD R,AC1          ; multiply the value in AC1 by constant R
        ADDD AC1,AC0        ; add this product to AC0
        STCDI AC0,(R1)      ; store the integer value at the output address
        LDFPS (SP)+         ; restore the floating point status
        RTS PC              ; return from subroutine
;
;-----------------------------------------------------------------------
;
;       DIFTOR has finished
;
;-----------------------------------------------------------------------
;
;       data allocation follows
;
;-----------------------------------------------------------------------
;
X1:     .FLT4   0.0         ; storage for the last input value
X2:     .FLT4   0.0         ; storage for the 2-nd last input value
X3:     .FLT4   0.0         ; storage for the 3-rd last input value
X4:     .FLT4   0.0         ; storage for the 4-th last input value
X5:     .FLT4   0.0         ; storage for the 5-th last input value
X6:     .FLT4   0.0         ; storage for the 6-th last input value
X7:     .FLT4   0.0         ; storage for the 7-th last input value
X8:     .FLT4   0.0         ; storage for the 8-th last input value
X9:     .FLT4   0.0         ; storage for the 9-th last input value
X10:    .FLT4   0.0         ; storage for the 10-th last input value
X11:    .FLT4   0.0         ; storage for the 11-th last input value
X12:    .FLT4   0.0         ; storage for the 12-th last input value
X13:    .FLT4   0.0         ; storage for the 13-th last input value
X14:    .FLT4   0.0         ; storage for the 14-th last input value
X15:    .FLT4   0.0         ; storage for the 15-th last input value
X16:    .FLT4   0.0         ; storage for the 16-th last input value
X17:    .FLT4   0.0         ; storage for the 17-th last input value
X18:    .FLT4   0.0         ; storage for the 18-th last input value
A:      .FLT4   0.008889            ; differentiator constant #1
B:      .FLT4   -0.01347            ; differentiator constant #2
C:      .FLT4   0.02680             ; differentiator constant #3
D:      .FLT4   -0.051667           ; differentiator constant #4
E:      .FLT4   0.092024            ; differentiator constant #5
F:      .FLT4   -0.15497            ; differentiator constant #6
G:      .FLT4   0.25667             ; differentiator constant #7
H:      .FLT4   -0.4462             ; differentiator constant #8
I:      .FLT4   0.97226             ; differentiator constant #9
J:      .FLT4   -0.97226            ; differentiator constant #10
K:      .FLT4   0.4462              ; differentiator constant #11
L:      .FLT4   -0.25667            ; differentiator constant #12
M:      .FLT4   0.15497             ; differentiator constant #13
N:      .FLT4   -0.092024           ; differentiator constant #14
O:      .FLT4   0.051667            ; differentiator constant #15
P:      .FLT4   -0.02680            ; differentiator constant #16
Q:      .FLT4   0.01347             ; differentiator constant #17
R:      .FLT4   -0.008889           ; differentiator constant #18
;
;-----------------------------------------------------------------------
;
;       end of MAC file
;
;-----------------------------------------------------------------------
;
        .END
```

D.7.24 The TDELAY subroutine

```
         .TITLE TDELAY
         .IDENT /X01/
         .GLOBL TDELAY
         .GLOBL $SAVAL
TDELAY:  JSR PC,$SAVAL
         MOV (R5)+,R0
         MOV (R5)+,R0
         MOV (R5),R1
         MOV X8,(R1)
         MOV X8,X9
         MOV X7,X8
         MOV X6,X7
         MOV X5,X6
         MOV X4,X5
         MOV X3,X4
         MOV X2,X3
         MOV X1,X2
         MOV (R0),X1
         RTS PC
X1:      .WORD
X2:      .WORD
X3:      .WORD
X4:      .WORD
X5:      .WORD
X6:      .WORD
X7:      .WORD
X8:      .WORD
X9:      .WORD
         .END
```

D.7.25 The PDFMOM subroutine

```
      SUBROUTINE PDFMOM(X,Y,N,IN,H)
      INTEGER N,IN,I,J,K
      REAL X(IN),Y(IN)
      REAL*8 H(6)
      INTEGER SYSCOR
      REAL RINTGL
      DO 10 I=1,N
      Y(I)=Y(I)*X(I)
10    CONTINUE
      H(1)=RINTGL(X,Y,N,IN,J)
      DO 20 I=1,N
      Y(I)=Y(I)*X(I)
20    CONTINUE
      H(2)=RINTGL(X,Y,N,IN,J)
      DO 30 I=1,N
      Y(I)=Y(I)*X(I)
30    CONTINUE
      H(3)=RINTGL(X,Y,N,IN,J)
      DO 40 I=1,N
      Y(I)=Y(I)*X(I)
40    CONTINUE
      H(4)=RINTGL(X,Y,N,IN,J)
      DO 50 I=1,N
      Y(I)=Y(I)*X(I)
50    CONTINUE
      H(5)=RINTGL(X,Y,N,IN,J)
      DO 60 I=1,N
      Y(I)=Y(I)*X(I)
```

```
60      CONTINUE
        H(6)=RINTGL(X,Y,N,IN,J)
        CALL STSCOR(H)
        RETURN
        END
```

D.7.26 The HP 7475 plotter subroutines

D.7.26.1 Subroutine PLTSET

```
C
C       This subroutine initializes the HP 7475 plotter to make a
C       plot.  It is only FORTRAN callable.
C
C       Calling conventions:     CALL PLTSET(CODE)
C
C       Argument:        CODE     - a numeric code for the paper size
C                                    1 - 8.5X11, X-axis on the long side
C                                    2 - 8.5X11, X-axis on the short side
C                                    3 - 11X17, X-axis on the long side
C                                    4 - 11X17, X-axis on the short side
C
C-----------------------------------------------------------------------
C
        SUBROUTINE PLTSET(CODE)
C
C       data declaration region
C
        INTEGER EMASK,CODE
C
C       EMASK   - an error mask
C       CODE    - the paper size code
C
C
C       initialize the plotter
C
        WRITE(4,100)
100     FORMAT(' ','IN;')
C
C       set the error code mask for all errors to be indicated
C
        EMASK=128+64+32+16+8+4+2+1
        WRITE(4,110) EMASK
110     FORMAT(' ','IM',I4,';')
C
C       set the paper size
C
C       Note: this subroutine does not check for errors in the paper size code
C
        GO TO(10,20,30,40) CODE
10      CONTINUE
        WRITE(4,120) 4
120     FORMAT(' ','PS',I4,';')
        WRITE(4,130) 0
130     FORMAT(' ','RO',I2,';IW;IP;')
        GO TO 50
20      CONTINUE
        WRITE(4,120) 4
        WRITE(4,130) 90
        GO TO 50
30      CONTINUE
        WRITE(4,120) 0
        WRITE(4,130) 0
        GO TO 50
40      CONTINUE
```

```
      WRITE(4,120) 0
      WRITE(4,130) 90
50    CONTINUE
C
C     this is the end of the subroutine
C
      RETURN
      END
```

D.7.26.2 Subroutine AXIS

```
C------------------------------------------------------------------------------
C
C     This subroutine draws the axis for a graph on the H-P 7475 plotter.
C     It is only FORTRAN callable.
C
C     Calling conventions:
C     CALL AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG,XLBL,IX,YLBL,IY)
C
C     ARGUMENTS:         XMIN   - the minimum X value
C                        XMAX   - the maximum X value
C                        YMIN   - the minimum Y value
C                        YMAX   - the maximum Y value
C                        CODE   - the paper size code
C                                  1 - 8.5X11, long side X-axis
C                                  2 - 8.5X11, short side X-axis
C                                  3 - 11X17, long side X-axis
C                                  4 - 11X17, short side X-axis
C                        YLOG   - logical, is the X-axis logarithmic
C                        YLOG   - logical, is the Y-axis logarithmic
C                        XLBL   - a 40 character X-axis lable array
C                        IX     - the number of characters in XLBL
C                        YLBL   - a 40 character Y-axis lable array
C                        IY     - the number of characters in YLBL
C
C--------------------------------------------------------------------------
C
      SUBROUTINE AXIS(XMIN,XMAX,YMIN,YMAX,CODE,XLOG,YLOG
     *   ,XLBL,IX,YLBL,IY)
C
C     data declaration region
C
      REAL XMIN,XMAX,YMIN,YMAX,DEL,XC,XB,YC,YB,DY,WIDTH,HEIGH
C
C     XMIN     - the minimum X value
C     XMAX     - the maximum X value
C     YMIN     - the minimum Y value
C     YMAX     - the maximum Y value
C     DEL      - a temporary storage value
C     YC       - the number of back shifts to print the Y label
C     YB       - the number of down shifts to print the Y label
C     XC       - the number of down shifts to print the X label
C     XB       - the number of back shifts to print the X label
C     WIDTH    - the width of the characters in cm.
C     HEIGH    - the height of the characters in cm.
C
      REAL NEG,POS,RISE,RUN,THETA
C
C     NEG      - the percentage of the graph size of the negative tick
C     POS      - the percentage of the graph size of the positive tick
C     RISE     - the rise of the direction of the characters
C     RUN      - the run of the direction of the characters
C     THETA    - the tangent of the slant angle of the characters
C
```

```
          INTEGER CODE,L,R,T,B,I,J,K,IX,IY
C
C         CODE      - the paper size code
C         L         - the left side of the graph
C         R         - the right side of the graph
C         T         - the top of the graph
C         B         - the bottom of the graph
C         I,J,K     - dummy indices
C         IX        - the number of characters in the X-axis lable
C         IY        - the number of characters in the Y-axis lable
C
          LOGICAL XLOG,YLOG,TERM,XLBL(40),YLBL(40)
C
C         XLOG      - is the X-axis logarithmic
C         YLOG      - is the Y-axis logarithmic
C         TERM      - the ACSII code for the label terminator
C         XLBL      - the X-axis lable array
C         YLBL      - the Y-axis lable array
C
C-------------------------------------------------------------------------
C
C         start of the code
C
C         set the label terminator to character number 3
C
          TERM=3
C
C         clear any previously set scaling parameters
C
          WRITE(4,400)
400       FORMAT(' ','SC;')
          GO TO (10,20,30,40) CODE
C
C         set points for the 8.5X11 paper with the X-axis on the long side
C
10        CONTINUE
          L=3200
          R=9296
          T=7620
          B=3048
          WIDTH=0.30
          HEIGH=0.45
          IF(IY.LT.12) GO TO 11
          IY=12
11        CONTINUE
          IF(IX.LT.33) GO TO 12
          IX=33
12        CONTINUE
          YC=-11.3
          YB=(FLOAT(IY-12)/2.0)+0.35-1.0
          XC=-5.6
          XB=(FLOAT(33-IX)/2.0)+0.435
          GO TO 50
C
C         set points for the 8.5X11 paper with the X-axis on the short side
C
20        CONTINUE
          L=2184
          R=6248
          T=9804
          B=2184
          WIDTH=0.20
          HEIGH=0.30
          IF(IY.LT.31) GO TO 21
          IY=31
21        CONTINUE
          IF(IX.LT.33) GO TO 22
          IX=33
22        CONTINUE
```

```
      YC=-12.7
      YB=(FLOAT(IY-31)/2.0)+0.375-1.0
      XC=-6.4
      XB=(FLOAT(33-IX)/2.0)+0.435
      GO TO 50
C
C     set points for the 11X17 paper with the X-axis on the long side
C
30    CONTINUE
      L=3200
      R=15392
      T=9804
      B=3708
      WIDTH=0.50
      HEIGH=0.75
      IF(IY.LT.10) GO TO 31
      IY=10
31    CONTINUE
      IF(IX.LT.40) GO TO 32
      IX=40
32    CONTINUE
      YC=-10.2
      YB=(FLOAT(IY-10)/2.0)+0.08-1.0
      XC=-5.1
      XB=(FLOAT(40-IX)/2.0)+0.32
      GO TO 50
C
C     set points for the 11X17 paper with the X-axis on the short side
C
40    CONTINUE
      L=3200
      R=9296
      T=16256
      B=3556
      WIDTH=0.30
      HEIGH=0.45
      IF(IY.LT.35) GO TO 41
      IY=35
41    CONTINUE
      IF(IX.LT.33) GO TO 42
      IX=33
42    CONTINUE
      YC=-11.3
      YB=(FLOAT(IY-35)/2.0)+0.14-1.0
      XC=-7.0
      XB=(FLOAT(33-IX)/2.0)+0.435
50    CONTINUE
C
C     set P1 and P2 to the respective corners of the graph
C
      WRITE(4,410) L,B,R,T
410   FORMAT(' ','IP',I5,',',I5,',',I5,',',I5,';')
C
C     select pen 1 (black , 0.7 mm. tip)
C
      WRITE(4,411) 1
411   FORMAT(' ','SP',I2,';')
C
C     select solid line type
C
      WRITE(4,412)
412   FORMAT(' ','LT;')
C
C     set the character size
C
      WRITE(4,413) WIDTH,HEIGH
413   FORMAT(' ','SI',F6.3,',',F6.3,';')
C
C     set the slant of the characters to be 0 degrees
C
      THETA=0.0
      WRITE(4,414) THETA
414   FORMAT(' ','SL',F6.3,';')
C
C     set the direction of the characters to 0 degrees
```

```
C
        RISE=0.0
        RUN=1.0
        WRITE(4,415) RUN,RISE
415     FORMAT(' ','DI',F6.3,',',F6.3,';')
C
C       lift the pen
C
        WRITE(4,420)
420     FORMAT(' ','PU;')
C
C       go to the lower left hand corner
C
        WRITE(4,430) L,B
430     FORMAT(' ','PA',I5,',',I5,';')
C
C       lower the pen
C
        WRITE(4,440)
440     FORMAT(' ','PD;')
C
C       draw the line from the lower left hand corner to the lower right
C
        WRITE(4,430) R,B
C
C       draw the line from the lower right hand corner to the top right
C
        WRITE(4,430) R,T
C
C       draw the line from the top right hand corner to the top left
C
        WRITE(4,430) L,T
C
C       draw the line from the top left hand corner to the lower left
C
        WRITE(4,430) L,B
C
C       lift the pen
C
        WRITE(4,420)
C
C       draw in the ticks and label the axis
C
C       X-axis (linear)
C
C
        IF(XLOG) GO TO 80
C
C       set the tick length to 0.5% of the graph length on
C       the positive side only
C
        POS=0.5
        NEG=0.0
        WRITE(4,450) POS,NEG
450     FORMAT(' ','TL',F5.1,',',F5.1,';')
C
C       draw a tick every 0.5 inches on the lower axis
C
        J=L+508
        K=R-508
        DO 60 I=J,K,508
        WRITE(4,430) I,B
        WRITE(4,460)
460     FORMAT(' ','XT;')
60      CONTINUE
C
```

```
C        lable the minimum and maximum X value
C
         WRITE(4,430) L,B
         WRITE(4,461)
461      FORMAT(' ','CP-3.0,-2.0;')
         WRITE(4,462) XMIN,TERM
462      FORMAT(' ','LB',F6.1,A1,';')
         WRITE(4,430) R,B
         WRITE(4,461)
         WRITE(4,462) XMAX,TERM
C
C        set the tick length to 0.5% of the graph length on
C        the negative side only
C
         WRITE(4,450) NEG,POS
C
C        draw a tick every 0.5 inches on the upper axis
C
         DO 70 I=J,K,508
         WRITE(4,430) I,T
         WRITE(4,460)
70       CONTINUE
         GO TO 100
C
C        X-axis (logarithmic)
C
80       CONTINUE
C
C        select the fine pen for the lines
C
         WRITE(4,411) 2
C
C        set the tick length to 100%  of the graph length on
C        the positive side only
C
         POS=100.0
         NEG=0.0
         WRITE(4,450) POS,NEG
C
C        find the number of X-axis cycles
C
         J=IFIX(XMAX-XMIN+0.99)
C
C        set IL to the minimum value's exponent
C
         IL=IFIX(XMIN)
C
C        compute the size of each cycle
C
         DEL=FLOAT(R-L)/FLOAT(J)
C
C        set the start to the left edge
C
         K=L
C
C        draw in the X ticks
C
         DO 90 I=1,J
         WRITE(4,430) K,B
         WRITE(4,460)
         WRITE(4,463) TERM
463      FORMAT(' ','SP1;CP-2.375,-3.0;LB10',A1,';CP-0.25,0.5;')
         WRITE(4,464) IL,TERM
464      FORMAT(' ','LB',I3,A1,';SP2;')
         IL=IL+1
C
```

```
C
C        add a value to k to move to the 2.0 position
         K=K+IFIX(DEL*ALOG10(2.0/1.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 3.0 position
C
         K=K+IFIX(DEL*ALOG10(3.0/2.0))
         WRITE(4,430) K,B
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 4.0 position
C
         K=K+IFIX(DEL*ALOG10(4.0/3.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 5.0 position
C
         K=K+IFIX(DEL*ALOG10(5.0/4.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 6.0 position
C
         K=K+IFIX(DEL*ALOG10(6.0/5.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 7.0 position
C
         K=K+IFIX(DEL*ALOG10(7.0/6.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 8.0 position
C
         K=K+IFIX(DEL*ALOG10(8.0/7.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 9.0 position
C
         K=K+IFIX(DEL*ALOG10(9.0/8.0))
         WRITE(4,430) K,B
         WRITE(4,460)
C
C        add a value to k to move to the 10.0 position
C
         K=K+IFIX(DEL*ALOG10(10.0/9.0))
90       CONTINUE
         WRITE(4,463) TERM
         WRITE(4,464) IL,TERM
100      CONTINUE
C
C        restore the thick pen
C
         WRITE(4,411) 1
         WRITE(4,430) L,B
         WRITE(4,560) XB,XC
560      FORMAT(' ','CP',F7.3,',',F7.3,';')
         DO 101 I=1,IX
         WRITE(4,570) XLBL(I),TERM
570      FORMAT(' ','LB',A1,A1,';')
```

```
 101      CONTINUE
 C
 C        Y-axis (linear)
 C
          IF(YLOG) GO TO 130
 C
 C        draw a tick every half inch
 C
          J=B+508
          K=T-508
 C
 C        set the tick length to 0.5% of the graph size on the positive
 C        side only
 C
          POS=0.5
          NEG=0.0
          WRITE(4,450) POS,NEG
          DO 110 I=J,K,508
          WRITE(4,430) L,I
          WRITE(4,470)
 470      FORMAT(' ','YT;')
 110      CONTINUE
 C
 C        label the minimum and maximum Y values
 C
          WRITE(4,430) L,B
          WRITE(4,471)
 471      FORMAT(' ','CP-7.0,-0.25;')
          WRITE(4,472) YMIN,TERM
 472      FORMAT(' ','LB',F6.1,A1,';')
          WRITE(4,430) L,T
          WRITE(4,471)
          WRITE(4,472) YMAX,TERM
 C
 C        set the tick length to 0.5% of the graph size on the negative
 C        side only
 C
          WRITE(4,450) NEG,POS
          DO 120 I=J,K,508
          WRITE(4,430) R,I
          WRITE(4,470)
 120      CONTINUE
          GO TO 150
 C
 C        Y-axis (logarithmic)
 C
 130      CONTINUE
 C
 C        select the thin pen for the lines
 C
          WRITE(4,411) 2
 C
 C        calculate the number of Y cycles
 C
          J=IFIX(YMAX-YMIN+0.99)
 C
 C        set IL to minimum value's exponent
 C
          IL=IFIX(YMIN)
 C
 C        calculate the size of each cycle
 C
          DEL=FLOAT(T-B)/FLOAT(J)
 C
 C        start from the bottom
 C
```

```
      K=8
C
C     set the tick length to 100% of the graph on the
C     positive side only
C
      POS=100.0
      NEG=0.0
      WRITE(4,450) POS,NEG
C
C     draw in each cycle
C
      DO 140 I=1,J
      WRITE(4,430) L,K
      WRITE(4,470)
      WRITE(4,491) TERM,IL,TERM
491   FORMAT(' ','SP1;CP-6.0,-0.5;LB10',A1
     *,';CP-0.25,0.5;LB',I3,A1,';CP2;')
      IL=IL+1
C
C     add a value to k to move to the 2.0 position
C
      K=K+IFIX(DEL*ALOG10(2.0/1.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
C     add a value to k to move to the 3.0 position
C
      K=K+IFIX(DEL*ALOG10(3.0/2.0))
      WRITE(4,430) L,K
      WRITF(4,470)
C
C     add a value to k to move to the 4.0 position

      K=K+IFIX(DEL*ALOG10(4.0/3.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
C     add a value to k to move to the 5.0 position
C
      K=K+IFIX(DEL*ALOG10(5.0/4.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
      add a value to k to move to the 6.0 position
C
      K=K+IFIX(DEL*ALOG10(6.0/5.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
C     add a value to k to move to the 7.0 position
C
      K=K+IFIX(DEL*ALOG10(7.0/6.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
C     add a value to k to move to the 8.0 position
C
      K=K+IFIX(DEL*ALOG10(8.0/7.0))
      WRITE(4,430) L,K
      WRITE(4,470)
C
C     add a value to k to move to the 9.0 position
C
      K=K+IFIX(DEL*ALOG10(9.0/8.0))
      WRITE(4,430) L,K
      WRITE(4,470)
```

```
C
C       add a value to k to move to the 10.0 position
C
        K=K+IFIX(DEL*ALOG10(10.0/9.0))
 140    CONTINUE
        WRITE(4,491) TERM,IL,TERM
 150    CONTINUE
C
C       restore the thick pen
C
        WRITE(4,411) 1
        WRITE(4,430) L,T
        WRITE(4,560) YC,YB
        IY=IY+3
        DO 151 I=1,IY
        WRITE(4,570) YLBL(I),TERM
        WRITE(4,560) -1.0,-1.0
 151    CONTINUE
C
C       set the input window
C
        WRITE(4,520) L,B,R,T
 520    FORMAT(' ','W',I5,',',I5,',',I5,',',I5,';')
C
C----------------------------------------------------------------------------
C
C       end of the code
C
C----------------------------------------------------------------------------
C
        RETURN
        END
```

D.7.26.3 Subroutine GRAPH

```
C--------------------------------------------------------------------------
C
C       This subroutine is FORTRAN callable.  It is used to plot a line
C       on the graph that has been set up by the RNGSHR, PLTSET and AXIS
C       subroutines.
C
C       Calling conventions:
C
C               CALL GRAPH(X,Y,N,IN,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
C
C
C       Arguments:
C
C               X       - the real array of the X coordinates
C               Y       - the real array of the Y coordinates
C               N       - the size of the X and Y arrays
C               IN      - the dimension of the X and Y arrays
C               XMIN    - the minimum value of the X coordinate to be plotted
C               XMAX    - the maximum value of the X coordinate to be plotted
C               YMIN    - the minimum value of the Y coordinate to be plotted
C               YMAX    - the maximum value of the Y coordinate to be plotted
C               LTYPE   - a line type code
C                               1  - dots at the points plotted
C                               2  - dot - space pattern
C                               3  - short dashes pattern
C                               4  - long dashes pattern
C                               5  - long dash - dot pattern
C                               6  - long dash - dashed pattern
C
```

```
C                                   7  - short dash - hyphen - hyphen pattern
C                                   8  - solid line
C                                   9  - no line, symbol mode
C                                   10 - solid line , symbol mode
C                   PEN      - the pen number ( only 1,2,3,4,5,6 permitted)
C                   SYM      - the ASCII code for the symbol
C
C-----------------------------------------------------------------------------
C
      SUBROUTINE GRAPH(X,Y,N,IN,XMIN,XMAX,YMIN,YMAX,LTYPE,PEN,SYM)
C
C------------------------------------------------------------------------------
C
C     data definition region
C
C-----------------------------------------------------------------------------
C
      INTEGER N,IN,LTYPE,PEN,I
C
C     N        - the size of the X and Y arrays
C     IN       - the dimension of the X and Y arrays
C     LTYPE    - the line type code
C     PEN      - the pen number
C     I        - dummy variable
C
      REAL X(IN),Y(IN),XMIN,XMAX,YMIN,YMAX,XSC,YSC,SX,SY
C
C     X        - the X coordinates array
C     Y        - the Y coordinates array
C     XMIN     - the minimum value of X to be plotted
C     XMAX     - the maximum value of X to be plotted
C     YMIN     - the minimum value of Y to be plotted
C     YMAX     - the maximum value of Y to be plotted
C     XSC      - the X coordinate scaling value
C     YSC      - the Y coordinate scaling value
C     SX       - the scaled X coordinate
C     SY       - the scaled Y coordinate
C

      LOGICAL SYM,LIFT
C
C     SYM      - the ASCII code for the symbol
C     LIFT     - the pen lift switch
C
C----------------------------------------------------------------------------
C
C     start of the code
C
C----------------------------------------------------------------------------
C
C     set the scaling points and the pen type
C
      WRITE(4,400) PEN
400   FORMAT(' ','SC0,1000,0,1000;SP',I2,';')
C
C     set the scaling factors
C
      XSC=1000.0/(XMAX-XMIN)
      YSC=1000.0/(YMAX-YMIN)
C
C     set the line type parameters
C
      GO TO (10,20,30,40,50,60,70,80,90,100) LTYPE
10    CONTINUE
```

```
C
C     set for a dot at each plotted point
C
      LIFT=.FALSE.
      WRITE(4,410) 0
410   FORMAT(' ','LT',I2,';')
      GO TO 110
20    CONTINUE
C
C     set for a dot - space pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 1
      GO TO 110
30    CONTINUE
C
C     set for a short dash pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 2
      GO TO 110
40    CONTINUE
C
C     set for a long dash pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 3
      GO TO 110
50    CONTINUE
C
C     set for a long dash - dot pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 4
      GO TO 110
60    CONTINUE
C
C     set for a long dash - hyphen pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 5

      GO TO 110
70    CONTINUE
C
C     set for a short dash - hyphen - hyphen pattern
C
      LIFT=.FALSE.
      WRITE(4,410) 6
      GO TO 110
80    CONTINUE
C
C     set for a solid line
C
      LIFT=.FALSE.
      WRITE(4,420)
420   FORMAT(' ','LT;')
      GO TO 110
90    CONTINUE
C
C     set for a symbol at each point
C
```

```
        LIFT=.TRUE.
        WRITE(4,430) SYM
430     FORMAT(' ','SM',A1,';')
        GO TO 110
100     CONTINUE
C
C       set for a solid line with a symbol at each point
C
        LIFT=.FALSE.
        WRITE(4,420)
        WRITE(4,430) SYM
110     CONTINUE
C
C       calculate the first scaled coordinate
C
        SX=XSC*(X(1)-XMIN)
        SY=YSC*(Y(1)-YMIN)
C
C       lift the pen
C
        WRITE(4,440)
440     FORMAT(' ','PU;')
450     FORMAT(' ','PD;')
C
C       move to the first point on the graph
C
        WRITE(4,460) SX,SY
460     FORMAT(' ','PA',F8.1,',',F8.1,';')
C
C       set the pen up or down depending on the lift switch
C
        IF(LIFT) GO TO 111
        WRITE(4,450)
111     CONTINUE
C
C       plot the graph
C
        DO 120 I=2,N
C
C       calculate the scaled coordinates
C
        SX=XSC*(X(I)-XMIN)
        SY=YSC*(Y(I)-YMIN)
C
C       move to the scaled coordinate
C

        WRITE(4,460) SX,SY
120     CONTINUE
C
C       lift the pen
C
        WRITE(4,440)
C
C--------------------------------------------------------------------------
C
C       end of the code, return from subroutine
C
C--------------------------------------------------------------------------
C
        RETURN
        END
```

D.7.26.4 Subroutine STRPEN

```
          SUBROUTINE STRPEN
          WRITE(4,400)
 400      FORMAT(' ','SP;')
          RETURN
          END
```

D.7.26.5 Subroutine RNGBEL

```
          SUBROUTINE RNGBEL(N)
          INTEGER I,N
          LOGICAL BEL
          BEL=7
          DO 10 I=1,N
          TYPE 500,BEL
 500      FORMAT('+', A1)
 10       CONTINUE
          RETURN
          END
```

Appendix E

Two Point Calculations

Attempts were made to calculate joint statistics such as the cross spectra and cross-correlations. However, it was found that because limitations in the bandwidth of the calculations due to memory limitations only a 2.7 decade FFT could be calculated. This resulted in what appeared to be a zero cross-spectrum except for one or two frequencies. The bandwidth could not be extended by decimation of the input stream without introducing very serious random sampling errors. Because the cross-spectra were too coarse, the resultant cross-correlations were unreliable. Hence no further discussion of these measurements will be given.

E.1 THE MOMENTS

Similar to the single point measurements, the input data streams were integrated for DC offset. They were then AC coupled using the subroutine FASAC described previously. Then a subroutine FASMOM was used to calculate all of the moments up to 4th order, including the means. This subroutine was a MACRO subroutine that converted the integer input stream into double precision floating point and accumulated the moments in double precision floating point. The resultant moments were corrected for any residual mean and were rounded to three decimal places.

E.2 THE JOINT PROBABILITY DENSITY FUNCTION

The input data streams were used to compute addresses in a 128X128 (16,384 points) array. This array was so large that it had to be declared as a virtual array and hence could not be easily called in subroutines. Because the array was virtual it had to be zeroed before it could be used to accumulate the joint probability density function histogram. To calculate the histogram, one was added to each element of the array for each occurance.

E.3 COMPUTER CODE

E.3.1 The two point moment program

```
C-----------------------------------------------------------------------
C
C       This program processes the data from two channels of the LPA II.
C       It accepts sampling rates up to the maximum that the A/D converter
C       can process.  The data is assumed to have been written on disk
C       drive 01 by the program FAST, which creates all of the necessary
C       files on this disk.
C
C       This program is used to calculate the results for the signals from
C       a X-array hot-wire anemometer.  All the moments up to the 4-th order
C       are calculated.  The results are written to a file on DK01: for other
C       programs to use.   Other quantities such as the join probability
C       distributions, the space-time correlation and the co- and quad-spectra
C       are calculated in other programs.
C
C       The samples from the second channel are assumed to fall between the
C       samples from the first channel.  Two new data sets are created, one
C       with the samples from channel one and another with data that has been
C       time delayed (by interpolation) to coincide with the samples of the
C       first channel.
C
C-----------------------------------------------------------------------
C
C       Data declaration and definition region
C
C-----------------------------------------------------------------------
C
```

```
      INTEGER BLK(2048),IONE(1024),ITWO(1024),I,J,L,ISIZE
C
C     BLK    -  a 2048 word buffer used to store the mixed input signal
C     IONE   -  a 1024 word buffer used to store the signal from channel one
C     ITWO   -  a 1024 word buffer used to store the signal from channel two
C     I,J,L  -  dummy variables
C     ISIZE  -  the number of mixed input signal buffers
C
      INTEGER ITAPE,ICHN1,ICHN2,IBAND
C
C     ITAPE  -  the tape number used to identify the data
C     ICHN1  -  a channel number used to identify the data set
C     ICHN2  -  a channel number used to identify the data set
C     IBAND  -  the band number used to identify the data
C
      INTEGER A(4),B(5),IOFF1,IOFF2,K1(2),K2(2)
C
C     A      - a 4 word buffer used to store the ascii code for the time
C     B      - a 5 word buffer used to store the ascii code for the date
C     IOFF1  - the integer value of the offset for channel 1
C     IOFF2  - the integer value of the offset for channel 2
C     K1     - a 2 word buffer used to compute the integral (sum) of the
C              signal from channel 1
C     K2     - a 2 word buffer used to compute the integral (sum) of the
C              signal from channel 2
C
      REAL*8 M1,M2,TEMP(14),MOMENT(14),T
C
C     M1       - the mean value of the signal from channel 1
C     M2       - the mean value of the signal from channel 2
C     TEMP     - a 14 (X4) word buffer used as a work butter
C     MOMENT   - a 14 (X4) word buffer used to accumulate and store the
C                computed moments
C     T        - a temperary storage value
C
      REAL DT,DC1,AC1,DC2,AC2
C
C     DT     - the sampling interval
C     DC1    - the DC part of the intial signal (m/sec,etc) for channel 1
C     AC1    - the AC part of the intial signal (m/sec,etc) for channel 1
C     DC2    - the DC part of the intial signal (m/sec,etc) for channel 2
C     AC2    - the AC part of the intial signal (m/sec,etc) for channel 2
C
C-------------------------------------------------------------------------------
C
C     Subroutine declarations
C
C-------------------------------------------------------------------------------
C
      INTEGER SLWDMP,SLWOPN,SLWCLS
C
C     SLWOPN- the subroutine that opens the data file to disk 01
C     SLWDMP- the subroutine that reads the data file
C     SLWCLS- the subroutine that closes the data file
C
      INTEGER BLKSFT,FASSUM,FASAC
C
C     BLKSFT- the subroutine that shifts a block of data from one location
C             location to another
C     FASSUM- the subroutine that integrates the data signal
C     FASAC - the subroutine that AC couples the signal
C
      REAL FASMOM,MOMCOR
C
C     FASMOM- the subroutine that calculates all the moments up to
C             order 4
C     MOMCOR- the subroutine that corrects the moments,computed by
C             FASMOM for offset error
C
```

```
      INTEGER TIME,DATE,IFIX
C
C     TIME  - the subroutine that returns the ascii code for the time
C     DATE  - the subroutine that returns the ascii code for the date
C     IFIX  - the subroutine that converts reals to integers
C
      REAL FLOAT
C
C     FLOAT - the subroutine that converts integers to reals
C
C-----------------------------------------------------------------------
C
C     Start of code
C
C-----------------------------------------------------------------------
C
C     get the present time and date
C
      CALL TIME(A)
      CALL DATE(B)
C
C     type the present time and date
C
      TYPE 1000,(A(I),I=1,4),(B(I),I=1,5)
1000  FORMAT(' ',80('*')/'0','Data processing commenced at '
     * ,4A2,' on ',5A2/' ')
C
C-----------------------------------------------------------------------
C
C     this section gets the digitization and signal information the was
C     saved by the program FAST
C
C-----------------------------------------------------------------------
C
C     open the information file that was written by FAST on disk DL0:
C
C     Note: logical unit number 1 must be assigned to DL1: when the
C           program is being task built.  Use ASN=DL1:1 in the options
C           section of TKB.
C
      OPEN(UNIT=1,TYPE='OLD',NAME='DL1:INFO.DAT')
C
C     read in the number of channels sampled,tape number and the band
C     number of the input signal
C
      READ(1,*) I,ITAPE,IBAND
C
C     if I does not equal 2 type a message and halt
C
      IF(I.NE.2) GO TO 70
C
C     read in the channel numbers
C
      READ(1,*) ICHN1,ICHN2
C
C     read in the intersample time period
C
      READ(1,*) DT
C
C     read in the the mean and RMS value of the signals
C
      READ(1,*) DC1,AC1
      READ(1,*) DC2,AC2
C
C     read in the number of 1024 word buffers
C
      READ(1,*) ISIZE
```

```
C
C       force ISIZE to be and even number
C
        ISIZE=2*(ISIZE/2)
C
C       close the information file
C
        CLOSE(UNIT=1,DISP='KEEP')
C
C       type the tape,channel and band numbers, and the number of 1024
C       word blocks of data on the terminal
C
        TYPE 1001,ITAPE,ICHN1,ICHN2,IBAND,ISIZE
1001    FORMAT(' ','For tape ',I2,'first channel ',I2,' second channel '
     *  ,I2,' band ',I2/' ','With ',I4,' buffers (1024 words) of data')
C
C       type the sampling interval and the mean and RMS values of the
C       input signal on the terminal
C
        TYPE 1002,DT,DC1,DC2,AC1,AC2
1002    FORMAT(' ','The input data was sampled every',3X,E10.3,' sec'/' '
     *  ,'The mean of the input signal was ',F8.3,' chn1 ',F8.3,
     *  ' chn2'/' ','The RMS of the input signal was  ',F8.3,' chn1 ',
     *  F8.3,' chn2'/'0')

C
C       open the data file to compute the integer off-sets
C
        CALL SLWOPN
C
C       intialize the mean values
C
        M1=0.0
        M2=0.0
C
C       integrate the signals to get the off-sets, ignoring the phase shift
C
        DO 20 I=1,ISIZE,2
C
C       fill  BLK with 64X32 words of data
C
        CALL SLWDMP(BLK(1),32)
        CALL SLWDMP(BLK(1025),32)
C
C       separate the signals into a channel one signal and a channel two signal
C
        L=1
        DO 10 J=1,1024
        IONE(J)=BLK(L)
        L=L+1
        ITWO(J)=BLK(L)
        L=L+1
10      CONTINUE
C
C       call FASSUM for each of the two buffers
C
        CALL FASSUM(IONE,1024,K1)
        CALL FASSUM(ITWO,1024,K2)
C
C       accumulate the mean values
C
        M1=M1+32767.0*FLOAT(K1(1))+FLOAT(K1(2))
        M2=M2+32767.0*FLOAT(K2(1))+FLOAT(K2(2))
20      CONTINUE
C
C       calculate the mean input values
C
        M1=M1/(FLOAT(ISIZE)*512.0)
        M2=M2/(FLOAT(ISIZE)*512.0)
C
C       compute the integer off-sets
C
```

```
      IOFF1=IFIX(M1)
      IOFF2=IFIX(M2)
C
C     calculate the mean input values in volts
C
      M1=(M1-2048.0)*2.44E-3
      M2=(M2-2048.0)*2.44E-3
C
C     type the mean input values and the integer offsets
C
      TYPE 1003,M1,IOFF1,M2,IOFF2
1003  FORMAT('0','Channel 1 ',F7.3,2X,'(Volts)  offset ',
     * 2X,I5/' ','Channel 2 ',F7.3,2X,'(Volts) offset ',
     * I5/'0')
C
C     reset the data file to read from the beginning
C
      CALL SLWCLS
C
C-----------------------------------------------------------------------
C
C     interpolate for channel 2 and compute the momoments
C
C-----------------------------------------------------------------------
C
C     zero the accumulator array MOMENT
C
      DO 25 I=1,14
      MOMENT(I)=0.0
25    CONTINUE
C
C     open the data file
C
      CALL SLWOPN
C
C     loop through the data
C
      DO 50 I=1,ISIZE,2
C
C     read in the first 2048 word buffer
C
      CALL SLWDMP(BLK(1),32)
      CALL SLWDMP(BLK(1025),32)
C
C     separate the buffer into data for channel 1 and channel 2
C
      L=1
      DO 30 J=1,1024
      IONE(J)=BLK(L)
      L=L+1
      ITWO(J)=BLK(L)
      L=L+1
30    CONTINUE
C
C     phase shift the data for channel 2
C
C     CALL PHSFT(ITWO,1024,DT,DWELL)
C
C     AC couple the signals
C
      CALL FASAC(IONE,1024,IOFF1)
      CALL FASAC(ITWO,1024,IOFF2)
C
C     compute the moments
C
      CALL FASMOM(IONE,ITWO,1024,TEMP)
C
C     accumulate the moments
C
      DO 40 J=1,14
      MOMENT(J)=MOMENT(J)+TEMP(J)
40    CONTINUE
50    CONTINUE
      T=2.0/FLOAT(ISIZE)
```

```
        DO 60 I=1,14
        MOMENT(I)=MOMENT(I)*T
60      CONTINUE
C
C       correct the moments for the residual error
C
        CALL MOMCOR(MOMENT)
C
C       convert the mean and RMS values to voltages
C
        MOMENT(1)=2.44E-3*MOMENT(1)
        MOMENT(2)=2.44E-3*MOMENT(2)
        MOMENT(5)=2.44E-3*MOMENT(5)
        MOMENT(6)=2.44E-3*MOMENT(6)
C
C       open the print file on logical unit 2
C
        OPEN(UNIT=2,NAME='PRINT.TXT',TYPE='NEW')
C
C       start a new page
C
        WRITE(2,2000)
2000    FORMAT('1')
C
C       write the present time and date
C
        WRITE(2,1000) (A(I),I=1,4),(B(I),I=1,5)
C
C       write the tape,channel and band number and the number of sample
C       buffers
C
        WRITE(2,1001) ITAPE,ICHN1,ICHN2,IBAND,ISIZE
C
C       write the sampling interval and the mean and RMS values to the
C       initial signals
C
        WRITE(2,1002) DT,DC1,DC1,AC1,AC2
C
C       write the computed moments
C
        WRITE(2,2001) MOMENT(1),MOMENT(5),MOMENT(2),MOMENT(6)
     *  ,MOMENT(3),MOMENT(7),MOMENT(4),MOMENT(8)
2001    FORMAT('0','Single channel moments'/'0',T15,'Channel 1',6X,
     *  'Channel 2'/' ','Mean offsets',4X,F9.5,6X,F9.5/' ','RMS',
     *  12X,F9.5,6X,F9.5,4X,'(Volts)'/' ','Skewness',7X,F9.5,6X,F9.5
     *  /' ','Flatness',7X,F9.5,6X,F9.5)
        WRITE(2,2002) (MOMENT(I),I=9,14)
2002    FORMAT('0','Mixed moments'/'0','Correlation coefficient',2X,
     *  F9.5/' ','(Ch. 1)X(Ch. 2)**2',7X,F9.5/' ','(Ch. 1)X(Ch. 2)**3'
     *  ,7X,F9.5/' ','(Ch.1)**2X(Ch. 2)',7X,F9.5/' ','((Ch. 1)X(Ch.'
     *  ,' 2))**2',5X,F9.5/' ','(Ch. 1)**3X(Ch. 2)',7X,F9.5/'0')
C
C       close the print file
C
        CLOSE(UNIT=2,DISP='PRINT')
        GO TO 80
70      CONTINUE
C
C       type an error message and halt
C
        TYPE 1004
1004    FORMAT(' The data file does not contain data for two channels',
     *  ', the request halted')
C
C       close the open data file
C
        CLOSE(UNIT=1,DISP='KEEP')
80      CONTINUE
C
C-----------------------------------------------------------------------
C
C       this is the end of the program
C
C-----------------------------------------------------------------------
```

```
C
      STOP
      END
```

E.3.2 The FASMOM subroutine

```
        .TITLE FASMOM
        .IDENT /FV01/
;
;-----------------------------------------------------------------------------
;
;       This is a Fortran or Macro callable subroutine.  FASMOM is used
;       to quickly calculate all of the moments up to fourth order for
;       two input data buffers.
;
;       Calling conventions:
;
;       Fortran:        CALL FASMOM(VEC1,VEC2,N,OUT)
;
;       Arguments:      VEC1    - an integer data array for channel 1
;                       VEC2    - an integer data array for channel 2
;                       N       - the integer size of the data arrays
;                       OUT     - an output buffer of 14. double precision
;                                 floating point values
;                                       #1  - mean of ch.1
;                                       #2  - mean square of ch.1
;                                       #3  - mean cube of ch.1
;                                       #4  - mean fourth of ch.1
;                                       #5  - mean of ch. 2
;                                       #6  - mean square of ch. 2
;                                       #7  - mean cube of ch. 2
;                                       #8  - mean fourth of ch. 2
;                                       #9  - correlation between ch.1 & ch. 2
;                                       #10 - mean of (ch. 1)*(ch. 2)**2
;                                       #11 - mean of (ch. 1)*(ch. 2)**3
;                                       #12 - mean of ((ch. 1)**2)*(ch. 2)
;                                       #13 - mean of ((ch. 1)**2)*(ch. 2**2)
;                                       #14 - mean of ((ch. 1)**3)*(ch. 2)
;
;-----------------------------------------------------------------------------
;
        AC0=%0  ;
        AC1=%1  ;
        AC2=%2  ; this section defines the floating point
        AC3=%3  ; accumulators
        AC4=%4  ;
        AC5=%5  ;
;
;-----------------------------------------------------------------------------
;
;       global definitions
;
;-----------------------------------------------------------------------------
;
        .GLOBL FASMOM   ; address of the FASMOM subroutine
        .GLOBL $SAVAL   ; address of the register saving subroutine
```

```
;
;-----------------------------------------------------------------------
;
;       start of FASMOM code
;
;-----------------------------------------------------------------------
;
FASMOM: JSR PC,$SAVAL     ; save all the registers
        STFPS -(SP)       ; save the floating point status
        MOV (R5)+,R0      ; get the number of arguments (unnessesary)
        MOV (R5)+,R0      ; get the starting address of ch. 1 data buffer
        MOV (R5)+,R1      ; get the starting address of ch. 2 data buffer
        MOV @(R5)+,R2     ; get the size of the data buffers
        MOV (R5)+,R3      ; get the starting address of the output buffer
        MOV R2,R4         ; save the size of the data buffer
        SETD              ; set floating double mode
        CLRD AC0          ; clear AC0
        STD AC0,U         ; clear storage location for ch. 1 (U) mean
        STD AC0,UU        ; clear storage location for U mean square
        STD AC0,UUU       ; clear storage location for U mean cube
        STD AC0,UUUU      ; clear storage location for U mean fourth
        STD AC0,V         ; clear storage location for ch. 2 (V) mean
        STD AC0,VV        ; clear storage location for V mean square
        STD AC0,VVV       ; clear storage location for V mean cube
        STD AC0,VVVV      ; clear storage location for V mean fourth
        STD AC0,UV        ; clear storage location for mean UV
        STD AC0,UVV       ; clear storage location for mean UV2
        STD AC0,UVVV      ; clear storage location for mean UV3
        STD AC0,UUV       ; clear storage location for mean U2V
        STD AC0,UUVV      ; clear storage location for mean U2V2
        STD AC0,UUUV      ; clear storage location for mean U3V
 1$:    LDCID (R0)+,AC0   ; load and convert a value from single word integer to
                          ;      double precision floating for ch. 1
        LDCID (R1)+,AC1   ; load and convert a value from single word integer to
                          ;      double precision floating for ch. 2
        STD AC0,AC4       ; store U in AC4
        STD AC1,AC5       ; store V in AC5
        LDD U,AC2         ; move the accumulated U to AC2
        LDD V,AC3         ; move the accumulated V to AC3
        ADDD AC0,AC2      ; add U to U accumulator
        STD AC2,U         ; save the accumulated U
        ADDD AC1,AC3      ; add V to V accumulator
        STD AC3,V         ; save the accumulated V
        MULD AC4,AC0      ; get square of U
        MULD AC5,AC1      ; get square of V
        LDD UU,AC2        ; move U2 to AC2
        LDD VV,AC3        ; move V2 to AC3
        ADDD AC0,AC2      ; accumulate square of U
        STD AC2,UU        ; save the accumulated U2
        ADDD AC1,AC3      ; accumulate square of V
        STD AC3,VV        ; save the accumulated U2
        MULD AC4,AC0      ; get the cube of U
        MULD AC5,AC1      ; get the cube of V
        LDD UUU,AC2       ; move U3 to AC2
        LDD VVV,AC3       ; move V3 to AC3
        ADDD AC0,AC2      ; accumulate cube of U
        STD AC2,UUU       ; save the accumulated U3
        ADDD AC1,AC3      ; accumulate cube of V
        STD AC3,VVV       ; save the accumulated V3
        MULD AC4,AC0      ; get fourth power of U
        MULD AC5,AC1      ; get fourth power of V
        LDD UUUU,AC2      ; move U4 to AC2
        LDD VVVV,AC3      ; move V4 to AC3
        ADDD AC0,AC2      ; accumulate fourth power of U
        ADDD AC1,AC3      ; accumulate fourth power of V
        STD AC2,UUUU      ; save the accumulated U4
        STD AC3,VVVV      ; save the accumulated V4
        LDD AC4,AC0       ; store U in AC0
        MULD AC5,AC0      ; get UV
```

```
LDD UV,AC1          ; move UV to AC1
ADDD AC0,AC1        ; accumulate UV
STD AC1,UV          ; save the accumulated UV
MULD AC5,AC0        ; set UV2
LDD UVV,AC1         ; move UV2 to AC1
ADDD AC0,AC1        ; accumulate UV2
STD AC1,UVV         ; save the accumulated UV2
MULD AC5,AC0        ; set UV3
LDD UVVV,AC1        ; move UV3 to AC1
ADDD AC0,AC1        ; accumulate UV3
STD AC1,UVVV        ; save the accumulated UV3
LDD AC4,AC0         ; store U in AC0
MULD AC4,AC0        ; set U2
MULD AC5,AC0        ; set U2V
LDD UUV,AC1         ; move U2V to AC1
ADDD AC0,AC1        ; accumulate U2V
STD AC1,UUV         ; save the accumulated U2V
MULD AC5,AC0        ; set U2V2
LDD UUVV,AC1        ; move U2V2 to AC1
ADDD AC0,AC1        ; accumulate U2V2
STD AC1,UUVV        ; save the accumulated U2V2
LDD AC4,AC0         ; store U in AC0
MULD AC4,AC0        ; set U2
MULD AC4,AC0        ; set U3
MULD AC5,AC0        ; set U3V
LDD UUUV,AC1        ; move U3V to AC1
ADDD AC0,AC1        ; accumulate U3V
STD AC1,UUUV        ; save the accumulated U3V
SUB #1,R2           ; subtract 1 from the size of the buffers
BGT 1$              ; branch to 1$ if there are more words left to process
LDCID R4,AC1        ; load and convert the size of the buffers into AC1
LDD U,AC0           ; recall U
DIVD AC1,AC0        ; normalize U
STD AC0,(R3)+       ; return U mean
LDD UU,AC0          ; recall U2
DIVD AC1,AC0        ; normalize U2
STD AC0,(R3)+       ; return U2
LDD UUU,AC0         ; recall U3
DIVD AC1,AC0        ; nomalize U3
STD AC0,(R3)+       ; return U3
LDD UUUU,AC0        ; recall U4
DIVD AC1,AC0        ; normalize U4
STD AC0,(R3)+       ; return U4
LDD V,AC0           ; recall V
DIVD AC1,AC0        ; normalize V mean
STD AC0,(R3)+       ; return V
LDD VV,AC0          ; recall V2
DIVD AC1,AC0        ; normalize V2
STD AC0,(R3)+       ; return V2
LDD VVV,AC0         ; recall V3
DIVD AC1,AC0        ; normalize V3
STD AC0,(R3)+       ; return V3
LDD VVVV,AC0        ; recall V4
DIVD AC1,AC0        ; normalize V4
STD AC0,(R3)+       ; return V4
LDD UV,AC0          ; recall UV
DIVD AC1,AC0        ; normalize UV
STD AC0,(R3)+       ; return UV
LDD UVV,AC0         ; recall UV2
DIVD AC1,AC0        ; normalize UV2
STD AC0,(R3)+       ; return UV2
LDD UVVV,AC0        ; recall UV3
DIVD AC1,AC0        ; normalize UV3
STD AC0,(R3)+       ; return UV3
```

```
          LDD UUV,AC0        ; recall U2V
          DIVD AC1,AC0       ; normalize U2V
          STD AC0,(R3)+      ; return U2V
          LDD UUVV,AC0       ; recall U2V2
          DIVD AC1,AC0       ; normalize U2V2
          STD AC0,(R3)+      ; return U2V2
          LDD UUUV,AC0       ; recall U3V
          DIVD AC1,AC0       ; normalize U3V
          STD AC0,(R3)+      ; return U3V
          LDFPS (SP)+        ; restore the floating point status
;
;------------------------------------------------------------------------------
;
;         this is the end of the code
;
;------------------------------------------------------------------------------
;
          RTS PC
;
;----------------------------------------------------------------------------
;
;         temporary data storage
;
;---------------------------------------------------------------------------
;
U:        .FLT4 0.0          ; accumulator for U
UU:       .FLT4 0.0          ; accumulator for U2
UUU:      .FLT4 0.0          ; accumulator for U3
UUUU:     .FLT4 0.0          ; accumulator for U4
V:        .FLT4 0.0          ; accumulator for V
VV:       .FLT4 0.0          ; accumulator for V2
VVV:      .FLT4 0.0          ; accumulator for V3
VVVV:     .FLT4 0.0          ; accumulator for V4
UV:       .FLT4 0.0          ; accumulator for UV
UVV:      .FLT4 0.0          ; accumulator for UV2
UVVV:     .FLT4 0.0          ; accumulator for UV3
UUV:      .FLT4 0.0          ; accumulator for U2V
UUVV:     .FLT4 0.0          ; accumulator for U2V2
UUUV:     .FLT4 0.0          ; accumulator for U3V
          .END
```

E.3.3 The MOMCOR subroutine

```
C-------------------------------------------------------------------------------
C
C         This subroutine is used to correct and normalize the moments that were
C         calculated in the subroutine FASMOM.  The subroutine compensates for
C         residual offset error.
C
C         calling convention:
C
C         Fortran:          CALL MOMCOR(ARRAY)
C
C         arguments:        ARRAY(1)  - the offset for channel 1
C                           ARRAY(2)  - the mean square of channel 1
C                           ARRAY(3)  - the mean cube of channel 1
C                           ARRAY(4)  - the fourth moment of channel 1
C                           ARRAY(5)  - the offset for channel 2
C                           ARRAY(6)  - the mean square of channel 2
```

```
C                         ARRAY(7)  - the mean cube of channel 2
C                         ARRAY(8)  - the fourth moment of channel 2
C                         ARRAY(9)  - the correlation between channel 1 and 2
C                         ARRAY(10) - the mean value of (ch 1)*(ch 2)**2
C                         ARRAY(11) - the mean value of (ch 1)*(ch 2)**3
C                         ARRAY(12) - the mean value of (ch 2)*(ch 1)**2
C                         ARRAY(13) - the mean value of ((ch 1)*(ch 2))**2
C                         ARRAY(14) - the mean value of (ch 2)*(ch 1)**3
C
C       output:           ARRAY     - The array is changed such that each
C                                     contains the corresponding central
C                                     moment non-dimesionalized by the RMS
C                                     values of each channel.   Except for
C                                     the offsets and the RMS values.
C
C-----------------------------------------------------------------------------
C
        SUBROUTINE MOMCOR(ARRAY)
C
C       data declaration region
C
        REAL*8 ARRAY(14)
C
C       ARRAY - a 14 (X4) word buffer containing the moments
C
C
C       the following are the correction to the moments to correct for the
C       residual offset error
C
        ARRAY(2)=ARRAY(2)-ARRAY(1)**2
        ARRAY(3)=ARRAY(3)-3.0*ARRAY(1)*ARRAY(2)-ARRAY(1)**3
        ARRAY(4)=ARRAY(4)-6.0*ARRAY(2)*(ARRAY(1)**2)
     *    -4.0*ARRAY(3)*ARRAY(1)-ARRAY(1)**4
        ARRAY(6)=ARRAY(6)-ARRAY(5)**2
        ARRAY(7)=ARRAY(7)-3.0*ARRAY(5)*ARRAY(6)-ARRAY(5)**3
        ARRAY(8)=ARRAY(8)-6.0*ARRAY(6)*(ARRAY(5)**2)
     *    -4.0*ARRAY(7)*ARRAY(5)-ARRAY(5)**4
        ARRAY(9)=ARRAY(9)-ARRAY(1)*ARRAY(5)
        ARRAY(10)=ARRAY(10)-2.0*ARRAY(9)*ARRAY(5)-ARRAY(6)*ARRAY(1)
     *    -ARRAY(1)*(ARRAY(5)**2)
        ARRAY(11)=ARRAY(11)-3.0*ARRAY(5)*ARRAY(10)-3.0*(ARRAY(5)**2)
     *    *ARRAY(9)-ARRAY(1)*ARRAY(7)-3.0*ARRAY(5)*ARRAY(1)*ARRAY(6)
     *    -ARRAY(1)*(ARRAY(5)**3)
        ARRAY(12)=ARRAY(12)-2.0*ARRAY(1)*ARRAY(9)-ARRAY(5)*ARRAY(2)
     *    -(ARRAY(1)**2)*ARRAY(5)
        ARRAY(13)=ARRAY(13)-2.0*ARRAY(5)*ARRAY(12)-(ARRAY(5)**2)*
     *    ARRAY(2)-2.0*ARRAY(1)*ARRAY(10)-4.0*ARRAY(1)*ARRAY(5)
     *    *ARRAY(9)-(ARRAY(1)**2)*ARRAY(6)-(ARRAY(1)*ARRAY(5))**2
        ARRAY(14)=ARRAY(14)-3.0*ARRAY(1)*ARRAY(12)-3.0*(ARRAY(1)**2)
     *    *ARRAY(9)-ARRAY(5)*ARRAY(3)-3.0*ARRAY(1)*ARRAY(5)*ARRAY(2)
     *    -(ARRAY(1)**3)*ARRAY(5)
C
C       this section normalizes the moments by the RMS values
C
        ARRAY(2)=SQRT(ARRAY(2))
        ARRAY(6)=SQRT(ARRAY(6))
        ARRAY(3)=ARRAY(3)/(ARRAY(2)**3)
        ARRAY(4)=ARRAY(4)/(ARRAY(2)**4)
        ARRAY(7)=ARRAY(7)/(ARRAY(6)**3)
        ARRAY(8)=ARRAY(8)/(ARRAY(6)**4)
        ARRAY(9)=ARRAY(9)/(ARRAY(2)*ARRAY(6))
        ARRAY(10)=ARRAY(10)/(ARRAY(2)*(ARRAY(6)**2))
        ARRAY(11)=ARRAY(11)/(ARRAY(2)*(ARRAY(6)**3))
        ARRAY(12)=ARRAY(12)/((ARRAY(2)**2)*ARRAY(6))
        ARRAY(13)=ARRAY(13)/((ARRAY(2)**2)*(ARRAY(6)**2))
        ARRAY(14)=ARRAY(14)/((ARRAY(2)**3)*ARRAY(6))
C
C       this section rounds the non-dimesional moments to the
C       nearest 0.1%
C
```

```
      ARRAY(3)=FLOAT(IFIX(1000.0*ARRAY(3)))/1000.0
      ARRAY(4)=FLOAT(IFIX(1000.0*ARRAY(4)))/1000.0
      ARRAY(7)=FLOAT(IFIX(1000.0*ARRAY(7)))/1000.0
      ARRAY(8)=FLOAT(IFIX(1000.0*ARRAY(8)))/1000.0
      ARRAY(9)=FLOAT(IFIX(1000.0*ARRAY(9)))/1000.0
      ARRAY(10)=FLOAT(IFIX(1000.0*ARRAY(10)))/1000.0
      ARRAY(11)=FLOAT(IFIX(1000.0*ARRAY(11)))/1000.0
      ARRAY(12)=FLOAT(IFIX(1000.0*ARRAY(12)))/1000.0
      ARRAY(13)=FLOAT(IFIX(1000.0*ARRAY(13)))/1000.0
      ARRAY(14)=FLOAT(IFIX(1000.0*ARRAY(14)))/1000.0
C
C     the subroutine has finished, return
C
      RETURN
      END
```

E.3.4 The joint probability density program

```
C-----------------------------------------------------------------------
C
C       This program processes the data from two channels of the LPA-11K.
C       It accepts sampling rates up to the maximum that the A/D converter
C       can process.  The data is assumed to have been written on disk
C       drive 01 by the program FAST, which creates all of the necessary
C       files on this disk.
C
C       This program is used to calculate the results for the signals from
C       a X-array hot-wire anemometer.  The joint probability densities are
C       computed.  Other quantities such as the moments and the spectra are
C       calculated in other programs.
C
C       The samples from the second channel are assumed to fall between the
C       samples from the first channel.  Two new data sets are created, one
C       with the samples from channel one and another with data that has been
C       time delayed (by interpolation) to coincide with the samples of the
C       first channel.
C
C-----------------------------------------------------------------------
C
C       Data declaration and definition region
C
C-----------------------------------------------------------------------
C
        VIRTUAL JDENS(128,128)
C
C       JDENS   - the virtual array containing the joint probability density
C
        INTEGER BLK(2048),IREAL1(1024),IREAL2(1024)
C
C       BLK     -  a 2048 word buffer used to store the mixed input signal
C       IREAL1  -  a 1024 word buffer used to store the signal from channel one
C       IREAL2  -  a 1024 word buffer used to store the signal from channel two
C
        INTEGER I,J,L,ISIZE,MAX1,MAX2,MIN1,MIN2
C
C       I,J,L -  dummy variables
C       ISIZE -  the number of mixed input signal buffers
C       MAX1  -  the maximum value of channel 1
C       MAX2  -  the maximum value of channel 2
C       MIN1  -  the minimum value of channel 1
C       MIN2  -  the mimimum value of channel 2
C
```

```
      INTEGER ITAPE,ICHN1,ICHN2,IBAND
C
C     ITAPE -  the tape number used to identify the data
C     ICHN1 -  a channel number used to identify the data set
C     ICHN2 -  a channel number used to identify the data set
C     IBAND -  the band number used to identify the data
C
      INTEGER A(4),B(5),IOFF1,IOFF2,K1(2),K2(2)
C
C     A     - a 4 word buffer used to store the ascii code for the time
C     B     - a 5 word buffer used to store the ascii code for the date
C     IOFF1 - the integer value of the offset for channel 1
C     IOFF2 - the integer value of the offset for channel 2
C     K1    - a 2 word buffer used to compute the integral (sum) of the
C             signal from channel 1
C     K2    - a 2 word buffer used to compute the integral (sum) of the
C             signal from channel 2
C
      REAL*8 M1,M2
C
C     M1      - the mean value of the signal from channel 1
C     M2      - the mean value of the signal from channel 2
C
      REAL DT,DC1,AC1,DC2,AC2,G1,G2
C
C     DT    - the sampling interval
C     DC1   - the DC part of the intial signal (m/sec,etc) for channel 1
C     AC1   - the AC part of the intial signal (m/sec,etc) for channel 1
C     DC2   - the DC part of the intial signal (m/sec,etc) for channel 2
C     AC2   - the AC part of the intial signal (m/sec,etc) for channel 2
C     G1    - the gain on channel 1
C     G2    - the gain on channel 2
C
      REAL JDENS
C
C     JDENS - the joint probabilaty density function
C
C-----------------------------------------------------------------------
C
C     Subroutine declarations
C
C-----------------------------------------------------------------------
C
      INTEGER SMLDMP,SMLOPN,SMLCLS
C
C     SMLOPN- the subroutine that opens the data file to disk or
C     SMLDMP- the subroutine that reads the data file
C     SMLCLS- the subroutine that closes the data file
C
      INTEGER FASSUM,FASAC
C
C     FASSUM- the subroutine that integrates the data signal
C     FASAC - the subroutine that AC couples the signal
C
      INTEGER TIME,DATE,IFIX
C
C     TIME  - the subroutine that returns the ascii code for the time
C     DATE  - the subroutine that returns the ascii code for the date
C     IFIX  - the subroutine that converts reals to integers
C
      REAL FLOAT
C
C     FLOAT - the subroutine that converts integers to reals
C
C-----------------------------------------------------------------------
C
C     Start of code
C
C-----------------------------------------------------------------------
C
C     get the present time and date
C
```

```
      CALL TIME(A)
      CALL DATE(B)
C
C     type the present time and date
C
      TYPE 1000,(A(I),I=1,4),(B(I),I=1,5)
1000  FORMAT(' ',80('*')/'0','Data processing commenced at '
     * ,4A2,' on ',5A2/' ')
C
C-----------------------------------------------------------------------
C
C     this section gets the digitization and signal information the was
C     saved by the program FAST
C
C-----------------------------------------------------------------------

C
C     open the information file that was written by FAST on disk DL1:
C
C     Note: logical unit number 1 must be assigned to DL1: when the
C           program is being task built.  Use ASN=DL1:1 in the options
C           section of TKB.
C
      OPEN(UNIT=1,TYPE='OLD',NAME='DL1:INFO.DAT')
C
C     read in the number of channels sampled,tape number and the band
C     number of the input signal
C
      READ(1,*) I,ITAPE,IBAND
C
C     if I does not equal 2 type a message and halt
C
      IF(I.NE.2) GO TO 90
C
C     read in the channel numbers
C
      READ(1,*) ICHN1,ICHN2
C
C     read in the intersample time period
C
      READ(1,*) DT
C
C     read in the the mean and RMS value of the signals
C
      READ(1,*) DC1,AC1
      READ(1,*) DC2,AC2
C
C     read in the number of 1024 word buffers
C
      READ(1,*) ISIZE
C
C     force ISIZE to be and even number
C
      ISIZE=2*(ISIZE/2)
C
C     close the information file
C
      CLOSE(UNIT=1,DISP='KEEP')
C
C     type the tape,channel and band numbers, and the number of 1024
C     word blocks of data on the terminal
C
      TYPE 1001,ITAPE,ICHN1,ICHN2,IBAND,ISIZE
1001  FORMAT(' ','For tape ',I2,' first channel ',I2,' second channel '
     * ,I2,' band ',I2/' ','With ',I4,' buffers (1024 words) of data')
C
C     type the sampling interval and the mean and RMS values of the
C     input signal on the terminal
C
```

```
      TYPE 1002,DT,DC1,DC2,AC1,AC2
1002  FORMAT(' ','The input data was sampled every',3X,E10.3,' sec'/' '
     * ,'The mean of the input signal was ',F8.3,' chn1 ',F8.3,
     * ' chn2'/' ','The RMS of the input signal was  ',F8.3,' chn1 ',
     * F8.3,' chn2'/'0')
C
C     open the data file to compute the integer off-sets
C
      CALL SMLOPN
C
C     intialize the mean values
C
      M1=0.0
      M2=0.0
C
C     initialize the minimum and maximum values for each channel
C
      MAX1=0
      MAX2=0
      MIN1=6000
      MIN2=6000
C
C     integrate the signals to get the off-sets, ignoring the phase shift
C
      DO 20 I=1,ISIZE,2
C
C     fill  BLK with 64X32 words of data
C
      CALL SMLDMP(BLK(1),32)
      CALL SMLDMP(BLK(1025),32)
C
C     separate the signals into a channel one signal and a channel two signal
C
      L=1
      DO 10 J=1,1024
      IREAL1(J)=BLK(L)
      L=L+1
      IREAL2(J)=BLK(L)
      L=L+1
10    CONTINUE
C
C     search for the maximum and minimum values for each channel
C
      DO 19 J=1,1024
      IF(MAX1.LT.IREAL1(J)) MAX1=IREAL1(J)
      IF(MAX2.LT.IREAL2(J)) MAX2=IREAL2(J)
      IF(MIN1.GT.IREAL1(J)) MIN1=IREAL1(J)
      IF(MIN2.GT.IREAL2(J)) MIN2=IREAL2(J)
19    CONTINUE
C
C     call FASSUM for each of the two buffers
C
      CALL FASSUM(IREAL1,1024,K1)
      CALL FASSUM(IREAL2,1024,K2)
C
C     accumulate the mean values
C
      M1=M1+32767.0*FLOAT(K1(1))+FLOAT(K1(2))
      M2=M2+32767.0*FLOAT(K2(1))+FLOAT(K2(2))
20    CONTINUE
C
C     calculate the mean input values
C
      M1=M1/(FLOAT(ISIZE)*512.0)
      M2=M2/(FLOAT(ISIZE)*512.0)
C
C     compute the integer off-sets
C
```

```
      IOFF1=IFIX(M1)
      IOFF2=IFIX(M2)
C
C     calculate the mean input values in volts
C
      M1=(M1-2048.0)*2.44E-3
      M2=(M2-2048.0)*2.44E-3
C
C     open the data file that will contain the joint probability
C     density data
C
      OPEN(UNIT=1,NAME='DL0:JDENS.DAT',TYPE='NEW')
C
C     write out the minimum and maximum voltages for each channel
C
      WRITE(1,*) MAX1,MAX2,MIN1,MIN2
C
C     write out the mean voltages and the integer offsets
C
      WRITE(1,*) M1,IOFF1,M2,IOFF2
C
C     type the mean input values and the integer offsets
C
      TYPE 1003,M1,IOFF1,M2,IOFF2
1003  FORMAT('0','Channel 1 ',F7.3,2X,'(Volts) offset ',
     * I5/' ','Channel 2 ',F7.3,2X,'(Volts) offset ',
     * I5/'0')
C
C     reset the data file to read from the beginning
C
      CALL SMLCLS
C
C-----------------------------------------------------------------------
C
C     set the RMS values for each channel
C
C-----------------------------------------------------------------------
C
C     zero M1 and M2 to accumulate the RMS values
C
      M1=0.0
      M2=0.0
C
C     open the data file
C
      CALL SMLOPN
C
C     loop through the data
C
      DO 50 I=1,ISIZE,2
C
C     read in the first 2048 word buffer
C
      CALL SMLDMP(BLK(1),32)
      CALL SMLDMP(BLK(1025),32)
C
C     separate the buffer into data for channel 1 and channel 2
C
      L=1
      DO 30 J=1,1024
      IREAL1(J)=BLK(L)
      L=L+1
      IREAL2(J)=BLK(L)
      L=L+1
30    CONTINUE
C
C     phase shift the data for channel 2
C
      CALL PHSFT(IREAL2,1024)
C
C     AC couple the signals
C
```

```
      CALL FASAC(IREAL1,1024,IOFF1)
      CALL FASAC(IREAL2,1024,IOFF2)
C
C     accumulate the mean square
C
      DO 40 J=1,1024
      M1=M1+FLOAT(IREAL1(J))**2
      M2=M2+FLOAT(IREAL2(J))**2
40    CONTINUE
50    CONTINUE
C
C     divide by the number of samples
C
      T=2.0/(FLOAT(ISIZE)*1024.0)
      M1=T*M1
      M2=T*M2
C
C     compute the RMS values of the channel 1 and channel 2 signals
C
      M1=SQRT(M1)
      M2=SQRT(M2)
C
C     convert the RMS values to integers
C
      IOFF1=IFIX(M1)
      IOFF2=IFIX(M2)
C
C     write the RMS values to the data file
C
      WRITE(1,*) M1,IOFF1,M2,IOFF2
C
C     calculate the RMS values in volts
C
      M1=2.44E-3*M1
      M2=2.44E-3*M2
C
C     type the RMS values to the terminal
C
      TYPE 1004,M1,IOFF1,M2,IOFF2
1004  FORMAT(' ',T20,'RMS Voltage',9X,'Integer value'/' ',
     *'Channel 1:',13X,F6.3,15X,I4/' ','Channel 2:',13X,
     *F6.3,15X,I4)
C
C     close the data file
C
      CALL SMLCLS
C
C     compute the JPDF
C
C
C     compute the scaling factors for the channels
C
      G1=4095.0/FLOAT(MAX1-MIN1)
      G2=4095.0/FLOAT(MAX2-MIN2)
C
C     zero the JPDF array
C
      DO 62 I=1,128
      DO 61 J=1,128
      JDENS(I,J)=0.0
61    CONTINUE
62    CONTINUE
C
C     reset the data file to the begining
C
      CALL SMLOPN
C
C     loop through the data
C
      DO 70 I=1,ISIZE,2
```

```
C
C       read in a 2048 word buffer of data
C
        CALL SMLDMP(BLK(1),32)
        CALL SMLDMP(BLK(1025),32)
C
C       fill in the JPDF array
C
C
C       loop through the buffer
C
        DO 60 J=1,2048,2
        L=J+1
C
C       compute the address of the array element
C
        I1=(IFIX(G1*FLOAT(BLK(J)-MIN1))/32)+1
        I2=(IFIX(G2*FLOAT(BLK(L)-MIN2))/32)+1
C
C       increment that array element
C
        JDENS(I1,I2)=JDENS(I1,I2)+1.0
60      CONTINUE
70      CONTINUE
C
C       close the data file
C
        CALL SMLCLS
C
C       write out the joint probability density data
C
        DO 80 I=1,128
        WRITE(1,*) (JDENS(I,J),J=1,128)
80      CONTINUE
C
C       close the output file
C
        CLOSE(UNIT=1,DISP='KEEP')
C
C       this is the end of the program
C
        GO TO 100
90      CONTINUE
C
C       type an error message and halt
C
        TYPE 1005
1005    FORMAT(' The data file does not contain data for two channels',
     *  ', the request halted')
C
C       close the open data file
C
        CLOSE(UNIT=1,DISP='KEEP')
100     CONTINUE
C
C-----------------------------------------------------------------------
C
C       this is the end of the program
C
C-----------------------------------------------------------------------
C
        STOP
        END
```

E.3.5 The JPDF plotting program

```
      LOGICAL SYM,TITLE(80)
      VIRTUAL JDENS(128,128)
      REAL JDENS
      REAL MEAN1,MEAN2,RMS1,RMS2,TOTAL
      REAL X(128),Y(128),MAX(128)
      REAL XOFF,XINC,YOFF,YINC,SHIFT,INC,M
      REAL HI1,HI2,LOW1,LOW2
      INTEGER I,J,K,L,I1,I2,I3,I4,PEN,LTYPE,CODE
      INTEGER MAX1,MAX2,MIN1,MIN2
      OPEN(UNIT=1,NAME='DL:JDENS.DAT',TYPE='OLD')
      READ(1,*) MAX1,MAX2,MIN1,MIN2
      READ(1,*) MEAN1,I1,MEAN2,I2
      READ(1,*) RMS1,I3,RMS2,I4
      DO 10 I=1,128
      READ(1,*) (JDENS(I,J),J=1,128)
      MAX(I)=0.0
10    CONTINUE
      CLOSE(UNIT=1,DISP='KEEP')
      OPEN(UNIT=1,NAME='DL:JPDFLBL.TXT',TYPE='OLD')
      READ(1,1000) (TITLE(I),I=1,40)
1000  FORMAT(40A1)
      READ(1,1000) (TITLE(I),I=41,80)
      CLOSE(UNIT=1,DISP='KEEP')
      HI1=(FLOAT(MAX1-I1))/RMS1
      LOW1=(FLOAT(MIN1-I1))/RMS1
      HI2=(FLOAT(MAX2-I2))/RMS2
      LOW2=(FLOAT(MIN2-I2))/RMS2
      MEAN1=FLOAT(MAX1-MIN1)/(128.0*RMS1)
      MEAN2=FLOAT(MAX2-MIN2)/(128.0*RMS2)
      SHIFT=0.0
      INC=2.0/64.0
      TOTAL=0.0
      M=0.0
      DO 30 I=1,128
      DO 20 J=1,128
      TOTAL=TOTAL+JDENS(I,J)
      IF(JDENS(I,J).LE.M) GO TO 15
      M=JDENS(I,J)
15    CONTINUE
20    CONTINUE
30    CONTINUE
      DO 50 I=1,128
      DO 40 J=1,128
      JDENS(I,J)=JDENS(I,J)/M
40    CONTINUE
50    CONTINUE
      M=M/(TOTAL*MEAN1*MEAN2)
      XOFF=-FLOAT(I1)/RMS1
      XINC=32.0/RMS1
      YOFF=FLOAT(4096-I2)/RMS2
      YINC=32.0/RMS2
      X(1)=XOFF
```

```
            DO 60 I=2,128
            X(I)=X(I-1)+XINC
60          CONTINUE
            TYPE 5000
5000        FORMAT(' Put an 8.5X11 paper in the plotter'/' ',
       *    'and type a carrage return when it is ready ',$)
            ACCEPT 5010
5010        FORMAT(' ')
            TYPE 5020,TOTAL,M
5020        FORMAT(' The data will be plotted for ',F8.0,
       *    ,' data points. With a maximum'/' ','joint '
       *    ,'probability density of ',E14.7,' .')
            TYPE 5030,LOW1,HI1,LOW2,HI2
 5030        FORMAT(' Channel 1 ranged from ',F7.4,' to ',F7.4,
        *    ' (S.D.)'/' ',' Channel 2 ranged from ',F7.4,
        *    ' to ',F7.4,' (S.D.)')
             CODE=2
             CALL PLTSET(CODE)
             CALL ISOAX(LOW1,HI1,LOW2,HI2,M,TITLE)
             PEN=2
             LTYPE=8
             SYM='*'
             DO 100 J=128,1,-2
             DO 90 I=1,128
             Y(I)=JDENS(I,J)+SHIFT
             IF(MAX(I).GT.Y(I)) GO TO 70
             MAX(I)=Y(I)
             GO TO 80
 70          CONTINUE
             Y(I)=MAX(I)
 80          CONTINUE
 90          CONTINUE
             SHIFT=SHIFT+INC
             CALL GRAPH(X,Y,128,128,X(1),X(128),0.0,3.0,LTYPE,PEN,SYM)
 100         CONTINUE
             CALL STRPEN
             STOP
             END
```

E.3.6 The ISOAX subroutine

```
              SUBROUTINE ISOAX(LOW1,HI1,LOW2,HI2,MAX,TITLE)
              LOGICAL LABLE(9),TITLE(80),ETX
              REAL LOW1,HI1,LOW2,HI2,MAX
              INTEGER L,R,T,B,M,N,O,I
              LABLE(1)='J'
              LABLE(2)=' '
              LABLE(3)='P'
              LABLE(4)=' '
              LABLE(5)='D'
              LABLE(6)=' '
              LABLE(7)='F'
              LABLE(8)=' '
              LABLE(9)='%'
              ETX=3
              MAX=MAX*100.0
              L=2032
              N=1219
              R=7112
              T=9220
              B=3124
              O=2920
```

```
      WRITE(4,400)
400   FORMAT(' ','SP1;')
      WRITE(4,401)
401   FORMAT(' ','IT;')
      WRITE(4,402)
402   FORMAT(' ','SI0.20,0.30;')
      WRITE(4,403)
403   FORMAT(' ','SL0.0;')
      WRITE(4,404)
404   FORMAT(' ','DI0.0,0.0;')
      WRITE(4,410)
410   FORMAT(' ','PU;')
      WRITE(4,420) L,T
420   FORMAT(' ','PA',I5,',',I5,';')
      M=T-(T-B)/3
      WRITE(4,430)
430   FORMAT(' ','PD;')
      WRITE(4,420) L,M
      WRITE(4,410)
      WRITE(4,440)
440   FORMAT(' ','TL0.0,0.5;')
      WRITE(4,450)
450   FORMAT(' ','YT;')
      M=M+(T-B)/12
      WRITE(4,420) L,M
      WRITE(4,450)
      M=M+(T-B)/12
      WRITE(4,420) L,M
      WRITE(4,450)
      M=M+(T-B)/12
      WRITE(4,420) L,M
      WRITE(4,450)
      WRITE(4,420) L,T
      WRITE(4,450)
      WRITE(4,460) MAX,ETX
460   FORMAT(' ','CP-6.0,-0.3;LB',F5.2,A1,';')
      M=T-(T-B)/6
      MAX=MAX/2.0
      WRITE(4,420) L,M
      WRITE(4,470) MAX,ETX
470   FORMAT(' ','CP-6.0,-0.3;LB',F5.2,A1,';')
      M=T-(T-B)/3
      WRITE(4,420) L,M
      WRITE(4,480) ETX
480   FORMAT(' ','CP-5.0,-0.3;LB0.00',A1,';')
      WRITE(4,420) L,T
      WRITE(4,481)
481   FORMAT(' ','CP-8.0,0.5;')
      DO 5 I=1,9
      WRITE(4,492) LABLE(I),ETX
5     CONTINUE
      M=T-(T-B)/3
      WRITE(4,410)
      WRITE(4,420) N,M
      WRITE(4,430)
      WRITE(4,420) N,B
      WRITE(4,410)
      WRITE(4,450)
      WRITE(4,420) N,M
      WRITE(4,450)
      M=IFIX(FLOAT(M-B)*ABS(HI2)/(ABS(LOW2)+ABS(HI2)))+B
      WRITE(4,420) N,M
      WRITE(4,450)
      WRITE(4,420) L,M
      WRITE(4,490) 0.0,ETX
      M=T-(T-B)/3
      WRITE(4,420) L,M
```

```
        WRITE(4,490) LOW2,ETX
490     FORMAT(' ','CP-15.0,-0.3;LB',F6.2,A1,';')
        WRITE(4,420) L,M
        WRITE(4,491)
491     FORMAT(' ','CP-16.0,-5;')
        LABLE(1)='C'
        LABLE(2)='h'
        LABLE(3)='a'
        LABLE(4)='n'
        LABLE(5)='n'
        LABLE(6)='e'
        LABLE(7)='1'
        LABLE(8)=' '
        LABLE(9)='2'
        DO 10 I=1,9
        WRITE(4,492) LABLE(I),ETX
492     FORMAT(' ','LB',A1,A1,';CP-1.0,-1.0;')
10      CONTINUE
        WRITE(4,420) L,B
        WRITE(4,490) HI2,ETX
        LABLE(9)='1'
        M=(L+R)/2
        WRITE(4,420) M,B
        WRITE(4,493) (LABLE(I),I=1,9),ETX
493     FORMAT(' ','CP-4.5,-4.0;LB',9A1,A1,';')
        WRITE(4,410)
        WRITE(4,420) L,0
        WRITE(4,451)
        WRITE(4,430)
        WRITE(4,420) R,0
        WRITE(4,410)
        WRITE(4,451)
451     FORMAT(' ','XT;')
        I=IFIX(FLOAT(R-L)*ABS(LOW1)/(ABS(LOW1)+ABS(HI1)))+L
        WRITE(4,420) I,0
        WRITE(4,451)
        WRITE(4,420) I,B
        WRITE(4,500) 0.0,ETX
        WRITE(4,420) R,0
        WRITE(4,420) L,B
        WRITE(4,500) LOW1,ETX
500     FORMAT(' ','CP-3.0,-2.0;LB',F6.2,A1,';')
        WRITE(4,420) R,B
        WRITE(4,500) HI1,ETX
        WRITE(4,420) L,B
        WRITE(4,510) (TITLE(I),I=1,40),ETX
510     FORMAT(' ','CP0.0,-8.0;LB',40A1,A1,';')
        WRITE(4,420) L,B
        WRITE(4,511) (TITLE(I),I=41,80),ETX
511     FORMAT(' ','CP0.0,-9.0;LB',40A1,A1,';')
        WRITE(4,520) L,B,R,T
520     FORMAT(' ','IP',I5,',',I5,',',I5,',',I5,';')
        WRITE(4,530) L,B,R,T
530     FORMAT(' ','IW',I5,',',I5,',',I5,',',I5,';')
        RETURN
        END
```