MEASUREMENT OF VOLUMETRIC BLOOD FLOW USING ULTRASOUND TIME-DOMAIN CORRELATION

BY

ILMAR ARTHUR HEIN

B.S., University of Illinois, 1981
M.S., University of Illinois, 1983

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 1991

Urbana, Illinois

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

THE GRADUATE COLLEGE

SEPTEMBER 1990

WE HEREBY RECOMMEND THAT THE THESIS BY

ILMAR ARTHUR HEIN

ENTITLED MEASUREMENT OF VOLUMETRIC BLOOD FLOW USING

ULTRASOUND TIME-DOMAIN CORRELATION

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR

THE DEGREE OF DOCTOR OF PHILOSOPHY

William D. O'Brien Jr. Director of Thesis Research

Timothy N. Trick Head of Department

Committee on Final Examination†

William D. O'Brien Jr. Chairperson

W. Kenneth Jenkins

Leon A. Frizzell

Ray Fisher

† Required for doctor's degree but not for master's.

O-517

MEASUREMENT OF VOLUMETRIC BLOOD FLOW USING ULTRASOUND TIME-DOMAIN CORRELATION

Ilmar Arthur Hein, Ph.D.
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign, 1991

The Ultrasound Time Domain Correlation (UTDC) technique can accurately and precisely estimate the volumetric fluid flow through a circular vessel without prior knowledge of the vessel diameter, flow velocity profile, or transducer measurement angle. This technique observes the change in arrival time (instead of frequency, as with Doppler techniques) of ultrasound reflected from scatterers. If a vessel with moving fluid is insonated by two ultrasonic pulses separated by time T, the scatterers within the ultrasonic beam will move some distance D between the first and second pulse. There will be a difference τ in the arrival times of the two pulses since the scatterer has moved. The scatterer velocity can be calculated from the time difference τ. By sampling the velocity at different positions along the vessel, the transducer measurement angle and the volumetric flow can be calculated.

Previous research has shown that this technique can estimate the volumetric flow rate of constant, fully developed laminar flow in a blood flow phantom with an accuracy of 15%. This dissertation represents continuing development of the UTDC technique towards the eventual goal of producing a clinical quantitative blood flowmeter. Blood flow phantom measurements have been made in conditions more closely resembling *in vivo* conditions, such as pulsatile flow, multiple vessels, different size vessels, and attenuation. A real-time UTDC blood flowmeter system interfaced to a commercial ultrasound imager has been constructed and *in vivo* measurements have been made in canines and humans.

ACKNOWLEDGEMENTS

I would like to most gratefully thank my mother and my father, without whose constant love, understanding, and support (both emotional and financial) I would never have made it this far; to Professor William D. O'Brien, Jr., for directing me towards this project, for giving me the opportunity to travel and present this work at different conferences, and for providing encouragement, patience, and funding; to Professor Jenkins and Mr. Jerome Chen, for design and construction of the RNS correlator; to Dr. James Zachary, without whose assistance the canine measurements would not have been possible; to sonographers Joyce Bender-Schmale and Roberta O'Connor, for assistance in locating blood vessels in the canines, as well as in humans, and for practical help in imaging peripheral blood flow; to Professor M. E. Clarke for technical assistance with the pulsatile pump; to Dr. Ray Fish, for assistance with medical aspects of the project; to Bob Cicone, Billy McNeil, and Wanda Elliot, for help in many technical (and non-technical) aspects of the project; and to Felice Chu, for help in construction of the A/D control logic circuit board.

I gratefully acknowledge the financial assistance from the National Institutes of Health and National Heart, Lung, and Blood Institute, grant HL39704; and for the donation of the MK500 ultrasound imager from Advanced Technology Laboratory, Incorporated.

TABLE OF CONTENTS

CHAPTER PAGE

1 INTRODUCTION. 1
- 1.1 Doppler Blood Flow Measurement Techniques. 1
- 1.2 Problems with Doppler. 2
- 1.3 Ultrasound Time-Domain Correlation Technique Development. 5
- 1.4 Thesis Outline 6

2 CURRENT METHODS OF MEASURING TISSUE MOTION BY ANALYSIS OF REFLECTED ULTRASOUND ECHOES. 8
- 2.1 Introduction 8
- 2.2 One-Dimensional Measurement Configurations. 10
  - 2.2.1 Cepstrum analysis. 10
    - 2.2.1.1 Power cepstrum. 11
    - 2.2.1.2 Complex cepstrum. 14
  - 2.2.2 Digital wall-tracking technique. 16
  - 2.2.3 Correlation techniques used to measure tissue motion. 21
    - 2.2.3.1 Estimating tissue motion from the value of a normalized correlation coefficient 21
    - 2.2.3.2 Velocity measurement by locating the maximum in the normalized correlation coefficient 28
    - 2.2.3.3 Estimating disturbedness of flow from the value of the correlation coefficient 30
    - 2.2.3.4 Maximum likelihood estimation 32
- 2.3 Two-Dimensional Measurement Configurations. 36
  - 2.3.1 Optical flow techniques . . .. .38
    - 2.3.1.1 Optical flow applied to echocardiography. 41
    - 2.3.1.2 Optical flow applied to assessment of contractility of skeletal muscle .45
  - 2.3.2 Measurement of blood flow from B-scan images .48
    - 2.3.2.1 Blood flow detection using Boolean operators 48
    - 2.3.2.2 Two-dimensional correlation search algorithm 51
    - 2.3.2.3 Sum-absolute-difference tracking algorithm 54
    - 2.3.2.4 Two-dimensional correlation search in the third moment domain. 55
- 2.4 Conclusion. 57

3 ULTRASOUND TIME-DOMAIN CORRELATION BLOOD FLOW MEASUREMENT TECHNIQUE 59
3.1 Time-Domain Velocity Measurement Concept 59
3.2 Reflection of Ultrasound by Blood Cell Scatterers. .61
3.3 Calculation of the Time Shift via Correlation. . . .63
3.4 One-Dimensional Velocity Measurements. 67
3.5 Two-Dimensional Velocity Measurements. 68
3.6 Weighted Axial Velocity Measurement. 71

4 BLOOD FLOW PHANTOM. 76
4.1 Flow Generation. .76
4.1.1 Constant flow generation. 76
4.1.2 Pulsatile flow generation 78
4.2 Flow Velocity Distribution in the Dialysis Tubing. .80
4.3 Independent Measure of Volumetric Flow 81
4.4 Blood Mimicking Fluid. 81
4.5 Other Modifications. 82

5 PREVIOUS SYSTEM MODIFICATIONS AND CONSTANT FLOW MEASUREMENTS. 83
5.1 Original Time-Domain Correlation System. 83
5.1.1 Replacement of the VAX. 84
5.1.2 Software modifications 84
5.1.2.1 Color flow images. 85
5.1.2.2 Three-dimensional perspective of flow 85
5.2 Constant Flow Experiments. 87
5.2.1 Non-symmetric flow measurement 87
5.2.2 Volumetric flow calculation by sum-of-differential flows 90
5.2.3 Multiple vessel experiment. 95
5.3 *In Vivo* Experiment. 98

6 PULSATILE FLOW MEASUREMENTS 102
6.1 Data Acquisition .102
6.2 Flow Determination 105
6.3 Results. 107
6.3.1 Volumetric flow results 107
6.3.2 Flow vs. time. 110
6.4 Discussion of Volumetric Flow Results 110
6.5 Discussion of Flow vs. Time Results. 117
6.6 Conclusion 118

7 TEMPORARY SYSTEM. 119
7.1 Temporary System Setup 119
7.1.1 Hardware acquisition of data from the sector display. 121
7.1.2 Software. .123
7.1.2.1 Data acquisition program 124
7.1.2.2 Correlation and velocity calculation program. 125
7.1.2.3 Graphics program 125
7.2 Temporary System Experiments 126
7.2.1 Blood flow phantom measurements 129
7.2.2 *In vivo* velocity measurements 133
7.2.2.1 Canine femoral vein measurement. . .135
7.2.2.2 Canine femoral artery measurement. .137
7.3 Summary. 139

8 REAL-TIME SYSTEM. 141
8.1 Real-Time System Setup 141
8.1.1 Acquisition of data from the sector display .143
8.1.2 Axial velocity measurement limits 143
8.1.3 UDA-RNC system. 144
8.1.3.1 A/D subsystem. 144
8.1.3.2 RNS correlator subsystem 145
8.1.3.3 Background of residue number system arithmetic. 145
8.1.4 Software. 150
8.1.4.1 UTDC flowmeter program 150
8.1.4.2 Correlation algorithm. 151
8.1.5 Speed of real-time system 152
8.2 Blood Flow Phantom Experiments 153
8.2.1 Validation with 6.5 mm dialysis tubing . . .153
8.2.2 Measurements at different power levels . . .155
8.2.3 Measurements with attenuating and reflecting media. 161
8.2.4 Effects of different number of echoes and Δ_{max}. 166
8.2.5 Multiple vessel experiments 168
8.2.6 Measurements in a 1.5 mm diameter tube. . . .175
8.3 *In vivo* Measurements 176
8.4 Conclusions. 184

REFERENCES. 186

APPENDIX A DETAILS OF PULSING UNIT 190

APPENDIX B PULSATILE PERIOD COUNTER SCHEMATIC. 191

APPENDIX C APPLE IIE INTERRUPT GENERATOR BOARD 193

APPENDIX D MODIFICATIONS TO ATL MK500 SYSTEM 194

APPENDIX E TEMPORARY SYSTEM PROGRAMS 200

APPENDIX F A/D SUBSYSTEM SCHEMATICS AND TIMING DIAGRAMS. 215

APPENDIX G UTDC BLOOD FLOWMETER PROGRAM 223

VITA. 248

CHAPTER 1

INTRODUCTION

The knowledge of the volumetric blood flow rate is an important quantity in the diagnosis of various diseases and trauma as well as in cardiovascular research. Volume blood flow is one of the best indicators of available oxygen and also of the ability of the heart to maintain normal body processes [1]. Classical methods of blood flow measurement, such as Fick's method, dye dilution, and angiographic techniques, are invasive and can be potentially harmful to the patient, particularly if the patient is already very ill [2]. Non-invasive blood flow measurement techniques are thus highly desirable.

1.1 Doppler Blood Flow Measurement Techniques

The use of ultrasound to non-invasively measure blood flow has become a standard clinical practice. The most popular ultrasonic measurement method is the well-known Doppler technique. Doppler techniques estimate the velocity of blood cell scatterers by analyzing the frequency change of ultrasound reflected from the moving blood cell scatterers. Mathematically, the Doppler equation can be written as

$$v = \frac{f_d\ c}{2\ f_t\ \cos(\theta)} \tag{1}$$

where f_t is the frequency of the transmitted signal, f_d is the Doppler shift (received frequency minus the transmitted frequency), c is the speed of sound in the medium, and θ is the angle between the sound beam and the axis of blood flow [3].

Currently there are two main Doppler ultrasound techniques: Continuous wave Doppler and Pulsed Doppler. Continuous wave Doppler techniques have excellent spatial average velocity flow determination capabilities but cannot measure velocity vs. range. Pulse Doppler transmits a broad-band ultrasonic burst into tissue and listens for the echo, and is capable of measuring velocity vs. range [1].

1.2 Problems with Doppler

The major drawback of Doppler techniques is that the Doppler shift f_d is never a single frequency but a band of frequencies with a spectrum similar to that for band-limited random noise [4]. This is illustrated in Figure 1, where the Doppler shift has been measured at different positions within a 7 mm tube with fully developed laminar flow [5]. The Doppler shift is not a single frequency, and the velocity information must somehow be estimated from the mean frequency of the frequency bands. Problems arise with Doppler measurements because there are many factors which can affect the shape of the Doppler frequency band

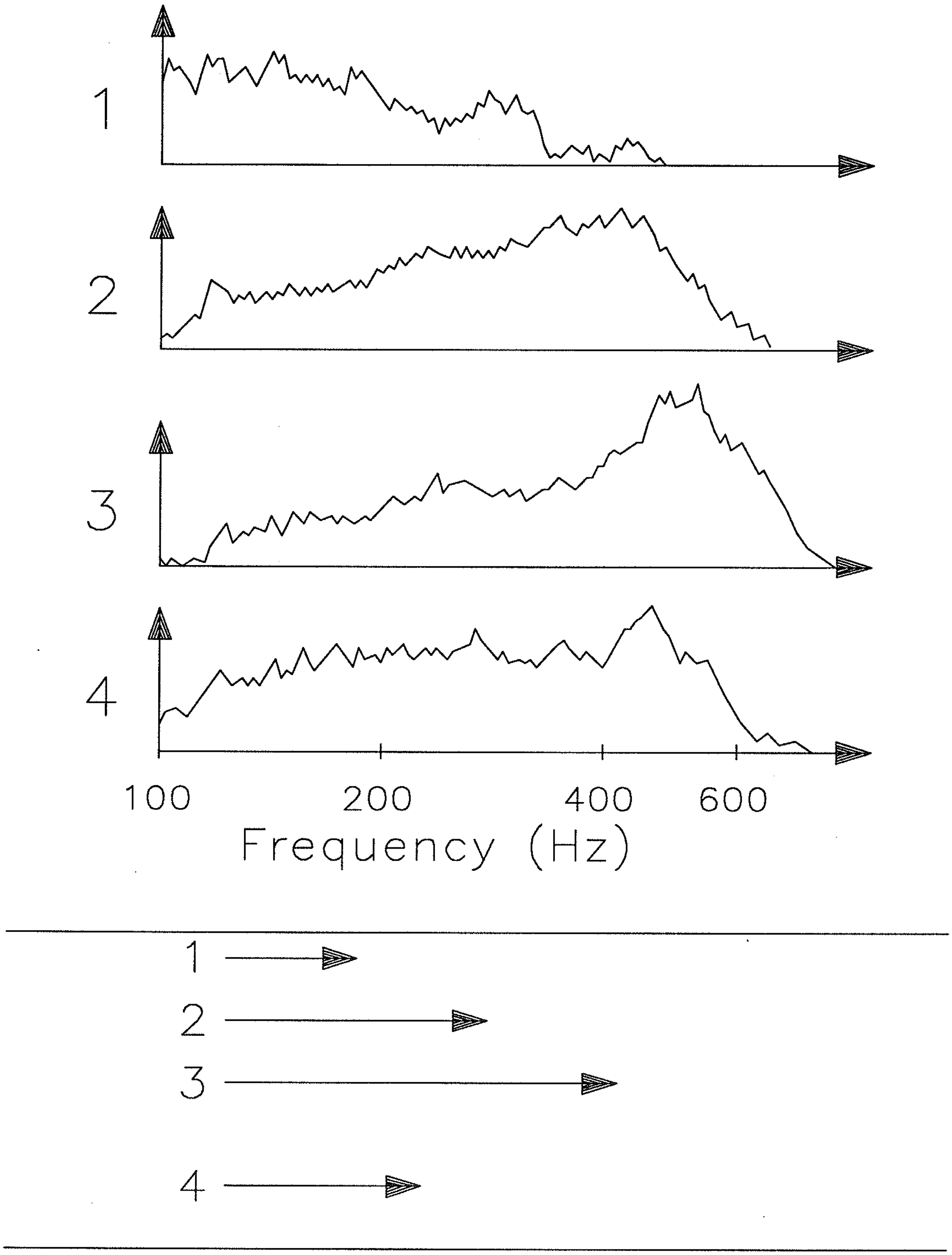


Figure 1 - Actual Doppler spectrum of flow velocities at selected positions along the diameter of a 7 mm tube. (from Olinger, [5])

that are not related to blood flow. A major factor is that of the frequency-dependent attenuation of tissues. The frequency-dependent total attenuation of tissue can be modeled as [6]:

$$A(f) = A_o f^b \tag{2}$$

where A(f) is the total attenuation as a function of frequency and A_o and b are frequency-independent tissue constants which are different for different tissues. The effect of frequency-dependent attenuation is to reduce the mean frequency of the Doppler frequency band [6].

Another frequency-dependent effect is that of backscattering. Moving blood is a group of particles which are much smaller than the wavelength of the ultrasound, and are normally considered to be Rayleigh scatterers. The scattered power is thus proportional to f^4, and higher frequencies will be scattered more than lower frequencies. Thus it somewhat counteracts the effects of tissue attenuation [6].

The problem with these frequency-dependent effects is that they are unpredictable. Because patients will vary tremendously in body composition, anywhere from muscular to obese, frequency-dependent affects will not be known beforehand.

Doppler also tends to have a poor signal to noise ratio. Since the ultrasonic burst is a time-gated sinusoid, much of the energy intended to be transmitted at the carrier frequency is lost into sidebands [7].

For these reasons, Doppler ultrasound tends to be more of a qualitative than quantitative blood flow measurement method. Recently, there has been a growing interest in using time-domain methods to measure blood flow. Time-domain methods have certain theoretical advantages over Doppler methods which potentially make them more accurate and precise than Doppler methods.

1.3 Ultrasound Time-Domain Correlation Technique Development

The Ultrasound Time-Domain Correlation (UTDC) technique has been under development at the Bioacoustics Research Laboratory at the University of Illinois for a number of years. Steve Foster developed the theory and conducted intensive computer simulation and error analysis of the UTDC technique, as well as one-dimensional experimental verification [7], [8]. Paul Embree developed a system capable of measuring continuous flow with a two-dimensional algorithm in a blood flow phantom, as well as calculating the theoretical precision of the technique [9], [10]. This dissertation represents the continuing development of the UTDC technique towards the eventual goal of producing a clinical quantitative blood flowmeter. Blood flow phantom measurements have been made in conditions more closely resembling *in-vivo* conditions, such as pulsatile flow, multiple vessels, different vessel sizes, and attenuation [11], [12]. A real-time system interfaced to a commercial ultrasound imager has been constructed and *in-vivo* measurements have been made in canines and humans.

1.4 Thesis Outline

The outline of this dissertation is as follows. Chapter 2 presents a literature review of different non-Doppler techniques used to assess tissue motion. Chapter 3 provides the background of the UTDC technique. Chapter 4 describes the blood flow phantom system used to validate the UTDC technique under various conditions. Chapter 5 describes modifications made to the system built by Embree [9] and presents continuous flow experiments made with multiple vessels and occluded vessels. The modified system used by Paul Embree is referred to as the "previous system." Chapter 6 validates the UTDC technique under pulsatile flow conditions. Chapter 7 introduces what is referred to as the "temporary system." This is a totally new system which uses a commercial ultrasound imager to locate vessels of interest and perform one-dimensional velocity vs. range measurements. It is referred to as the temporary system because the UTDC technique has been implemented entirely in software, and hence was not real time. This system was used to make *in-vivo* measurements in canine subjects while the final real-time system was under construction.

Chapter 8 describes the final product of this dissertation, the real-time system. This system performs the same measurements as the temporary system, except that the correlations are performed in hardware in real time with a custom-designed residue number system correlator. This system is evaluated under conditions with multiple vessels, with attenuating and scattering material between the vessel and transducer, under different signal-to-noise ratios, and with smaller diameter vessels. Measurements in the human carotid have also been made.

CHAPTER 2

CURRENT METHODS OF MEASURING TISSUE MOTION BY ANALYSIS OF REFLECTED ULTRASOUND ECHOES

This chapter presents an overview of different non-Doppler ultrasonic techniques used to measure tissue motion. It is a literature survey of methods currently under development by different research institutions which have the potential to be clinically useful in diagnosing various medical disorders.

2.1 Introduction

Ultrasound has become a standard clinical tool in the diagnosis and treatment of illness and injury. Perhaps the most well-known use of ultrasound is in the noninvasive imaging of different parts of the body. Ultrasound does, however, have other applications in medical diagnosis, such as in tissue characterization and measurement of tissue motion *in vivo*. The measurement of tissue motion is a broad category. It can include the analysis of the motion of physically active organs such as the heart and discrete structures such as cardiac valves and arterial walls. It can be used to measure motion in soft tissues (such as the liver) due to the pulsing of an artery in the tissue. The soft tissue motion is a function of the elasticity of the tissue, which is directly related to the state of the tissue (diseased or healthy). Finally, the measurement of blood flow velocity is an important parameter in diagnosing coronary diseases such as venous thrombosis.

Regardless of the particular medical application of ultrasound, all applications require the transmission and capture of radio-frequency (RF) ultrasonic signals. After capture, the signals must be processed in some way to extract the desired information. In the case of measuring tissue motion, many methods are currently under development. This literature survey has limited its scope to non-Doppler techniques for two reasons. First of all, Doppler techniques have been around for a long time, and have already been extensively covered in the literature. The application of (non-Doppler) time-domain techniques in clinical medicine, however, is more recent and has not been comprehensively reviewed. Second, time-domain methods appear to have certain theoretical advantages over Doppler methods which result in greater accuracy and repeatability. These advantages will be discussed where applicable.

The techniques presented in this overview include cepstrum analysis, correlation techniques, and optical flow methods. They have been grouped into two general categories: one-dimensional and two-dimensional measurement configurations. All of the techniques, except for cepstrum analysis, are purely time-domain techniques.

2.2 One-Dimensional Measurement Configurations

A one-dimensional measurement configuration consists of measurements made along a single transducer A-line. The ultrasound transducer is pointed in a given direction and data are acquired along this line and processed accordingly.

2.2.1 Cepstrum analysis

The cepstrum of a function is defined as the inverse Fourier transform of the logarithm of the Fourier transform of that function. Mathematically, if $F(\omega)$ is the Fourier transform of a function f(t), the cepstrum can be expressed as a complex cepstrum or a power cepstrum: [13]:

Complex cepstrum: $$C(\tau) = F^{-1}\left[\log F(\omega)\right] \tag{3}$$

Power Cepstrum: $$C(\tau) = \left\{F^{-1}\left[\log\ |F(\omega)|^{2}\right]\right\}^{2} \tag{4}$$

where F^{-1} represents the inverse Fourier transform. The power cepstrum can be used to estimate the time separation in two similar ultrasound pulse waveforms, and the complex cepstrum can be used to detect periodically vibrating structures.

2.2.1.1 Power cepstrum

Figure 2 illustrates how the power cepstrum can be used to estimate the spacing between two reflective surfaces for the unlimited bandwidth case [14]. If a transducer transmits a pulse towards two surfaces, then each surface will reflect an echo. The difference in arrival times of the two echoes is directly proportional to the physical distance between the reflectors. In this example, they are displaced in time by 2 μs. Figure 2a) shows the time-domain signal received by the transducer. The first echo arrives 1 μs after transmission, and the 2nd echo 2 μs after the first echo. Figure 2b) shows the power spectral density of the waveform; Figure 2c) shows the log-power spectral density; and Figure 2d) shows the power cepstrum. The units of the cepstrum are "quefrency," which is not strictly time but rather difference in time. Thus the cepstrum has a peak at a quefrency of 2 μs, which corresponds to the difference in time between reception of echoes reflected from the surfaces. It also contains "rahmonics," which are similar to harmonics, due to multiple reflections between the reflectors.

The advantage of using power cepstrum analysis is shown in Figure 3 [14]. All ultrasonic signals are of limited bandwidth, and if the reflectors are closely spaced together, it may become difficult to separate the echoes in the time domain. Figure 3a) shows the case where the reflectors are far enough apart such that the two echoes are still distinguishable. If the reflectors are

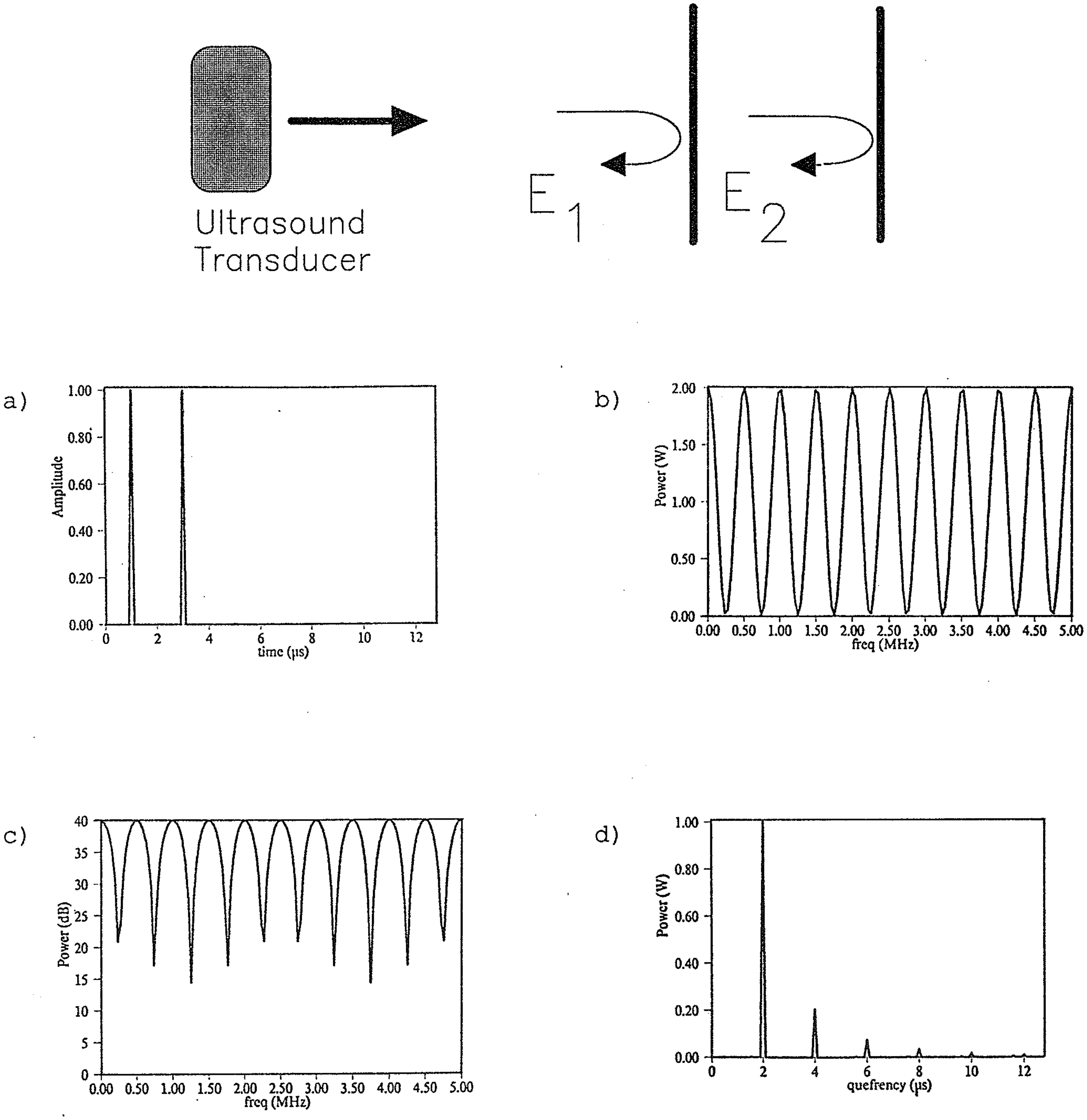


Figure 2 - Case of two reflectors and unlimited bandwith waveform. a) Reflected signal waveform. b) Power spectral density. c) Log-power spectral density. d) Power cepstrum. (From Kuc, Haghkerdar, O'Donnell, [14])

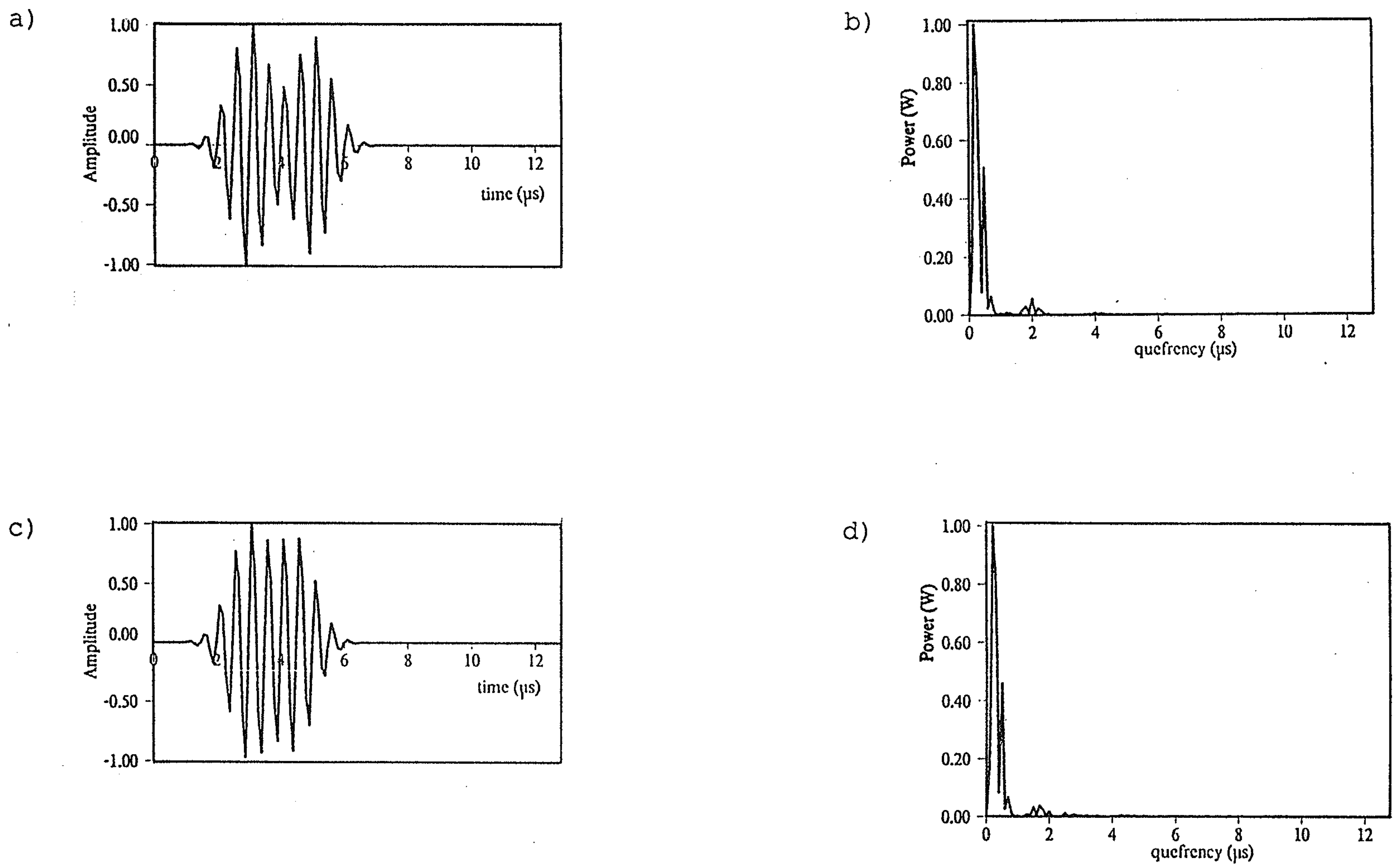


Figure 3 - a) Reflected signal for two reflectors and a bandlimited waveform. b) The power cepstrum. c) Reflected signal for two very closely spaced reflectors. It is difficult to distinguish one echo from the other. d) The power cepstrum still has a distinguishable peak. (From Kuc, Haghkerdar, and O'Donnell [14])

moved closer together, however, the two echoes become undistinguishable from each other, as shown in Figure 3c). In the cepstrum domain, however, a distinct peak still exists for both cases (Figures 3b) and 3d)). In biological situations, the reflectors will be different tissue interfaces or structures, and one use of power cepstrum is to detect periodicities in tissue structures [15]. In some cases the tissue will be moving, and tissue motion can be detected by the changing power cepstrum.

2.2.1.2 Complex cepstrum

The complex cepstrum can be used to detect vibrating structures, as illustrated in Figure 4 [16]. In this figure, an ultrasound transducer transmits an ultrasound signal u_t:

$$u_t(t) = U_t \cos(2\pi f_t t) \tag{5}$$

where f_t is the transmitted frequency and U_t is the amplitude of the transmitted signal. If the motion of the structure can be described as $A\sin(2\pi f_m t)$, where A is the amplitude and f_m is the frequency of the vibration, the received ultrasound signal $u_r(t)$ can be expressed as

$$U_r(t) = U_r \cos\left\{ 2\pi f_t + 2\left[\{x + A\sin(2\pi f_m t)\}2\pi f_t/c \right]\right\} \tag{6}$$

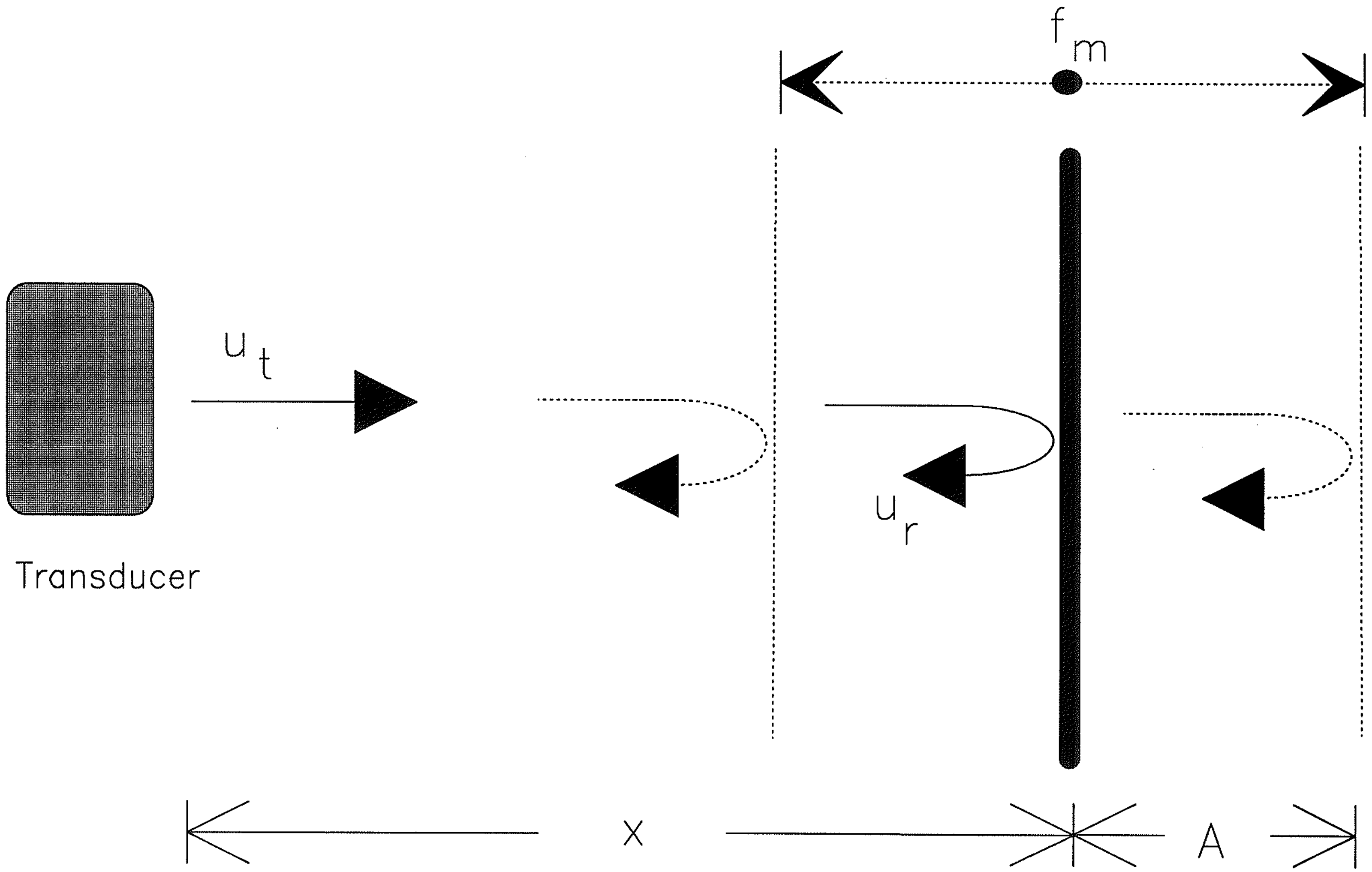


Figure 4 - Reflection of an ultrasound signal by a vibrating structure at a distance x from the transducer. The structure vibrates with a displacement of Asin($2\pi f_m$ t). (Adapted from Nowicki and Marciniak, [16])

where x is the mean displacement of the vibrating structure from the transducer and c is the speed of sound. The demodulated received signal will be

$$u_d = \cos\left[m_f \sin(2\pi f_m t)\right] \tag{7}$$

where $m_f = 2A2\pi f_t/c$. If the complex cepstrum of u_d is calculated according to Equation (3), the frequency of vibration of the structure can be isolated. This is illustrated in Figure 5. The spectrum of the demodulated (Figure 5a)) signal will contain ripples in the frequency band spaced by Δf, where $\Delta f = 2f_m$. The cepstrum (Figure 5b)) will contain a peak located at $T = 1/\Delta f$, where $f_m = 1/(2T)$ [16].

This technique is useful in the detection of vibrations of blood vessel walls or heart valves, whose typical frequency of vibration is under 100 Hz. Doppler ultrasound cannot be used to detect these vibrations since the typical cutoff frequency of diagnostic Doppler is approximately 100 Hz. Nowicki and Marciniak [16] have successfully used this technique to diagnose innocent heart murmurs in young children.

2.2.2 Digital wall-tracking technique

Perhaps the simplest method of measuring tissue motion with ultrasound echoes is to track the rising edge of an echo reflected from a tissue interface, such as a vessel wall. Powalowski [17]

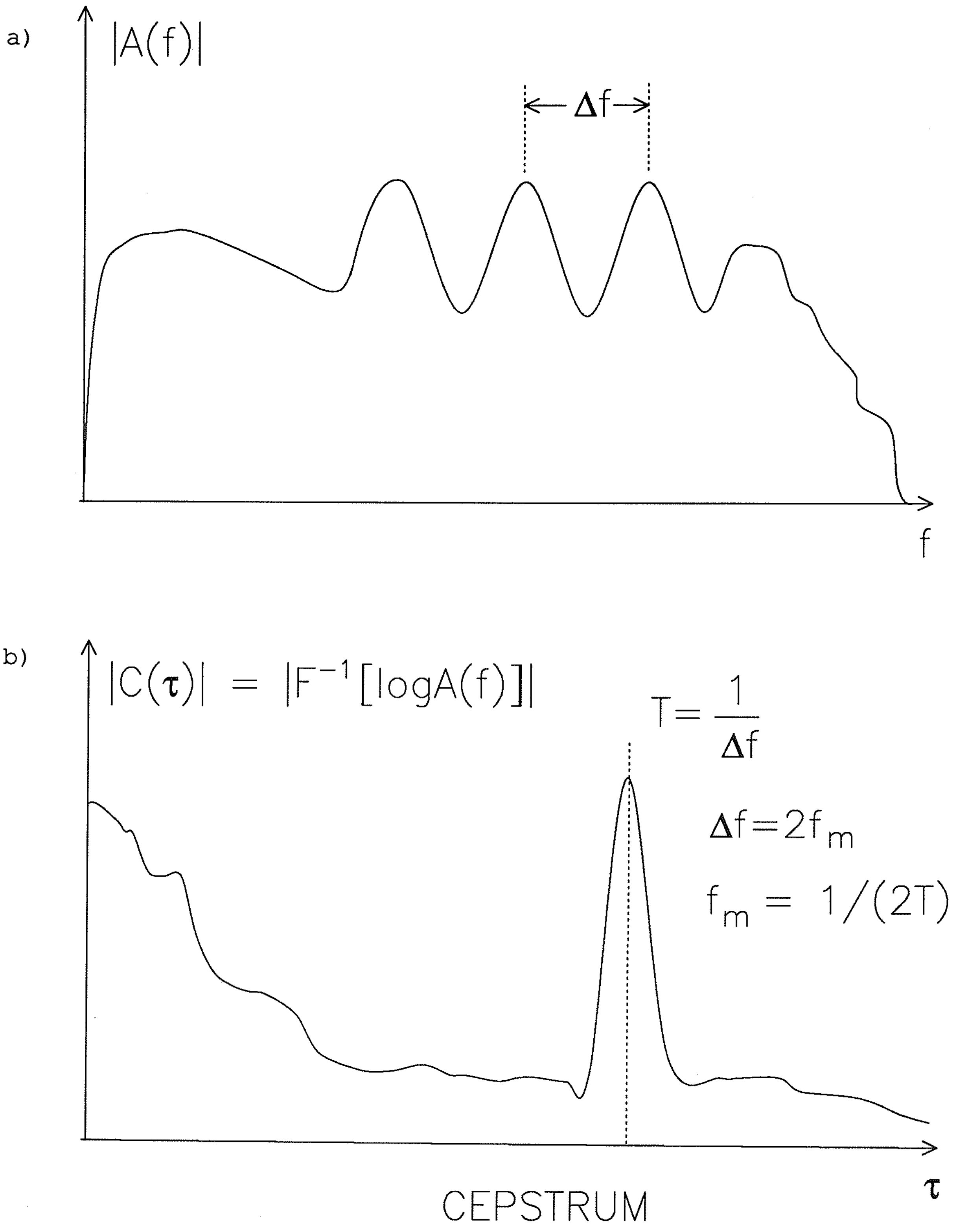


Figure 5 - a) Spectrum and b) Cepstrum of the demodulated ultrasound signal reflected from a periodically vibrating surface. (From Nowicki and Marciniak, [16])

has described such a technique, illustrated in Figure 6. Here a pulsed ultrasound transducer is directed at a vessel and the reflections from the vessel are recorded. The transducer is pulsed at a rate of T_p (Figure 6a)) and echoes E1 and E2 are received from the interior and exterior surfaces of a vessel wall (Figure 6b)). If the analog waveforms are fed into a comparator circuit, the comparator will produce an output pulse whenever the analog waveform exceeds the trigger level, as in Figure 6c). The digital pulses from the comparator will move back and forth in time along with the motion of the vessel wall. It is a simple matter to digitally track this motion using a tracing gate and a clock, as in Figures 6d) and 6e). To identify a desired echo slope, the tracing gate is generated before it. Clock pulses are counted from the transmission of the pulse until the first rising echo edge after the tracing gate. The use of this digital wall tracking technique has been used by Powalowski [17] to measure the hemodynamic properties of blood vessels. His system is illustrated in Figure 7. It consists of two ultrasound transducers, P1 and P2, where P1 is a pulsed 6.75 MHz transducer which is used to measure the vessel wall diameter, and P2 is a 4.5 MHz continuous wave transducer used to measure the blood flow velocity. If the instantaneous vessel wall diameter and blood flow velocities are known, various hemodynamic parameters such as the volumetric blood flow rate, blood pressure, input vessel impedance, vessel diameter change over work cycle of the heart, pulse wave velocity, and coefficient of rigidity of the blood

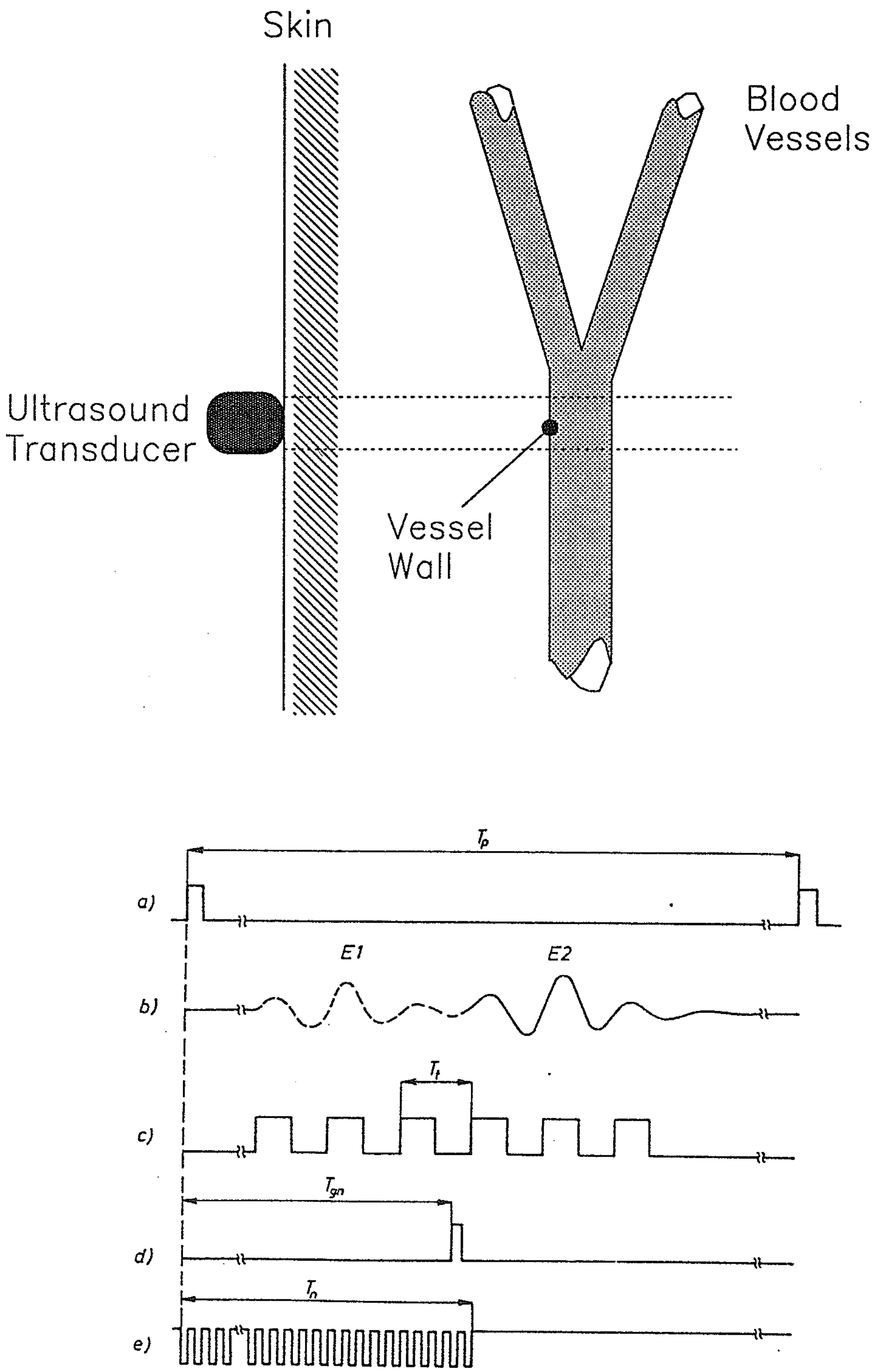


Figure 6 - The measurement of the time variable position of the vessel wall: a) trigger impulse, b) echoes detected from the external and internal blood vessel walls, c) echoes at output of level comparator, d) tracing gate, e) clock impulses. (from Powalowski, [17])

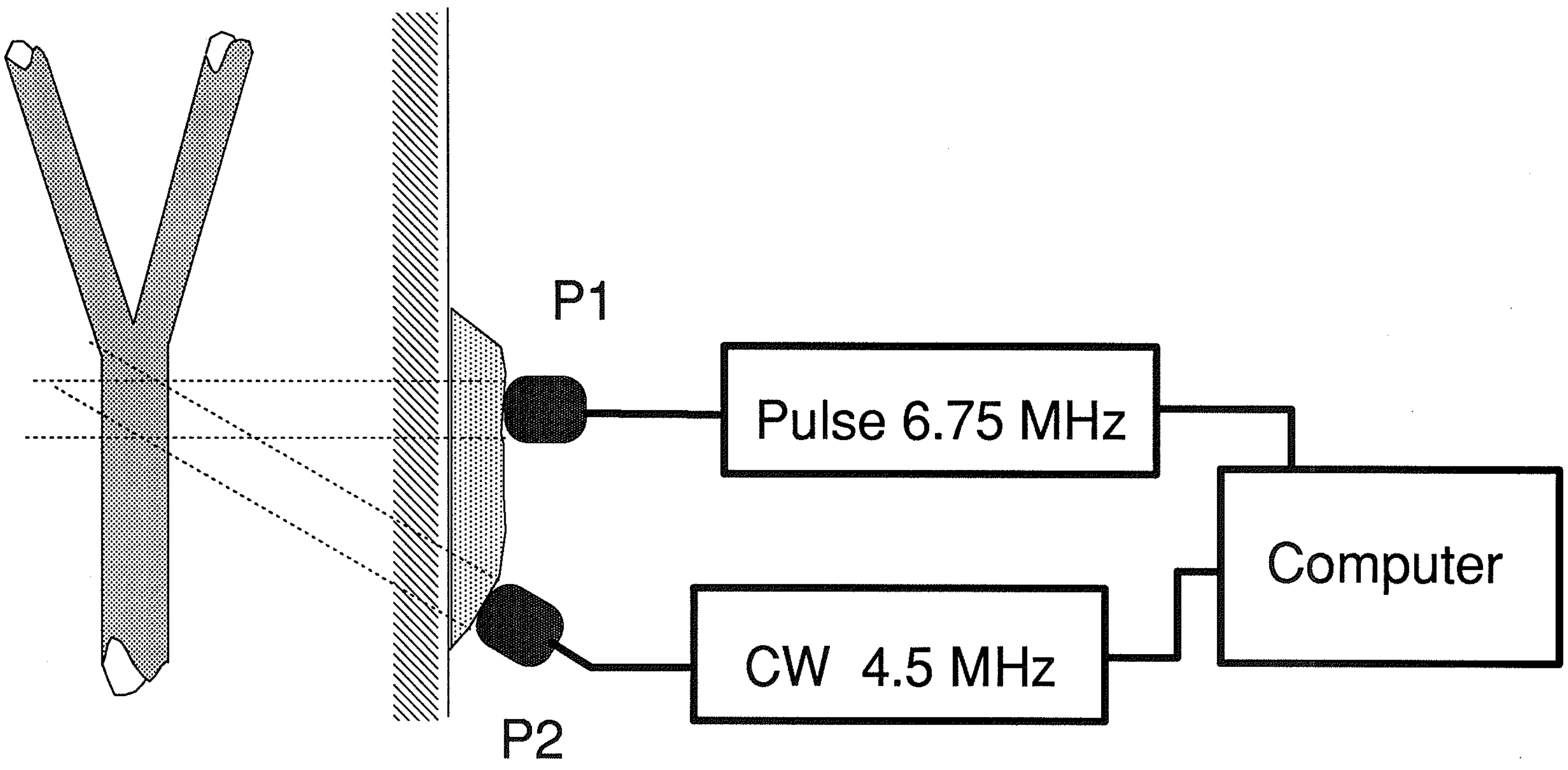


Figure 7 - Hemodynamic measurement system. P1 measures vessel wall diameter and P2 simultaneously measures blood flow velocity. With wall diameter and blood flow information, the computer can calculate the blood flow rate, blood pressure, input vessel impedance, vessel diameter change over work cycle of the heart, pulse wave velocity, and coefficient of rigidity of the blood vessel wall. (From Powalowski, [17])

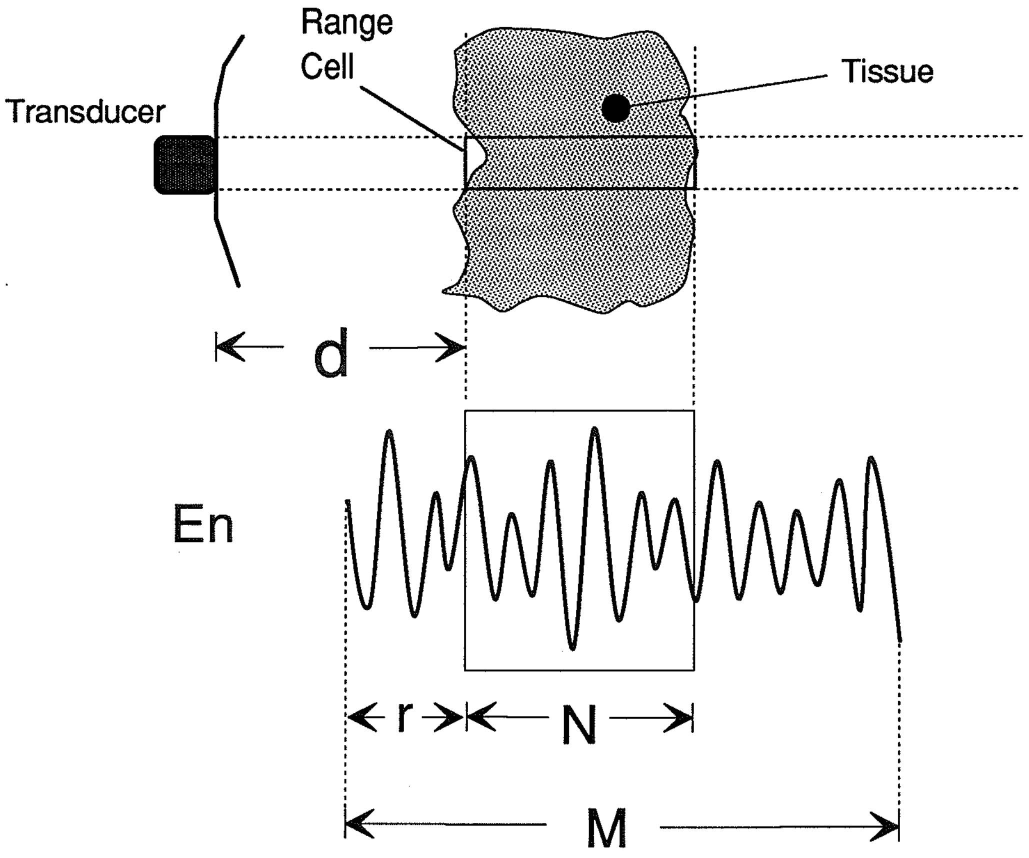
vessel walls can be estimated. These parameters are important in the diagnosis of many circulatory diseases such as arteriosclerosis.

2.2.3 Correlation techniques used to measure tissue motion

The digital wall tracking technique has serious limitations since it only observes rising edges of ultrasound echoes. The tracing gate must be set precisely, and if the walls move out of the range of the tracing gate, there could be ambiguities as to what tissues are under observation. A more accurate method of measuring tissue motion is to track a segment of echo reflected from a group of scatterers rather than just the rising edge. This can be accomplished using various correlation techniques.

2.2.3.1 Estimating tissue motion from the value of a normalized correlation coefficient

Figure 8 illustrates how correlation can be used to estimate the motion within a range cell at a distance d from the transducer. If an ultrasound pulse is transmitted, then an echo will be reflected by tissues in the transducer beam. Each area of tissue will have an ultrasound footprint, the shape of which depends on the structure of the particular tissue. The physical width of the range cell will correspond to a window of digital length N within the total digitized echo E_n (total length M), and the location of the footprint within the echo depends on the



$$R = \frac{\sum_{i=0}^{N-1} E1(r+i)\ E2(r+i)}{\sqrt{\sum_{j=0}^{N-1} [\ E1(r+j)\ ]^{2} \sum_{k=0}^{N-1} [\ E2(r+k)\ ]^{2}}}$$

Figure 8 - Tracking a section of tissue via correlation. The normalized correlation coefficient R can be calculated for a window of length N at a distance d from the transducer for two digitized ultrasound echoes.

distance the tissue scatterers are from the transducer. If more than one echo is digitized and stored, the normalized correlation coefficient can be calculated for the window length N and distance d from the transducer:

$$R = \frac{\sum_{i=0}^{N-1} E_1(r+i)\,E_2(r+i)}{\sqrt{\sum_{j=0}^{N-1} [E_1(r+j)]^2 \sum_{k=0}^{N-1} [E_2(r+k)]^2}} \tag{8}$$

where r is the location within the echo corresponding to the distance d. The normalized correlation coefficient R is basically a value between minus one and one which indicates how similar the two windowed echoes are. If they are exactly the same, then R=1, and if they have no similarity, then R=0. Thus the amount of tissue motion between transmission of ultrasonic pulses can be estimated from the value of the normalized correlation coefficient calculated between different echoes. The more the tissue has moved, the smaller the value of R. This is illustrated in Figure 9 for tissue motion moving axially away from the transducer. In this case three pulses are transmitted and three echoes received. If the tissue is moving with a velocity v, then the tissue will move a distance d_1 between the 1st and 2nd echo and d_2 between the 1st and 3rd echo. In this example the correlation coefficient R_{12}

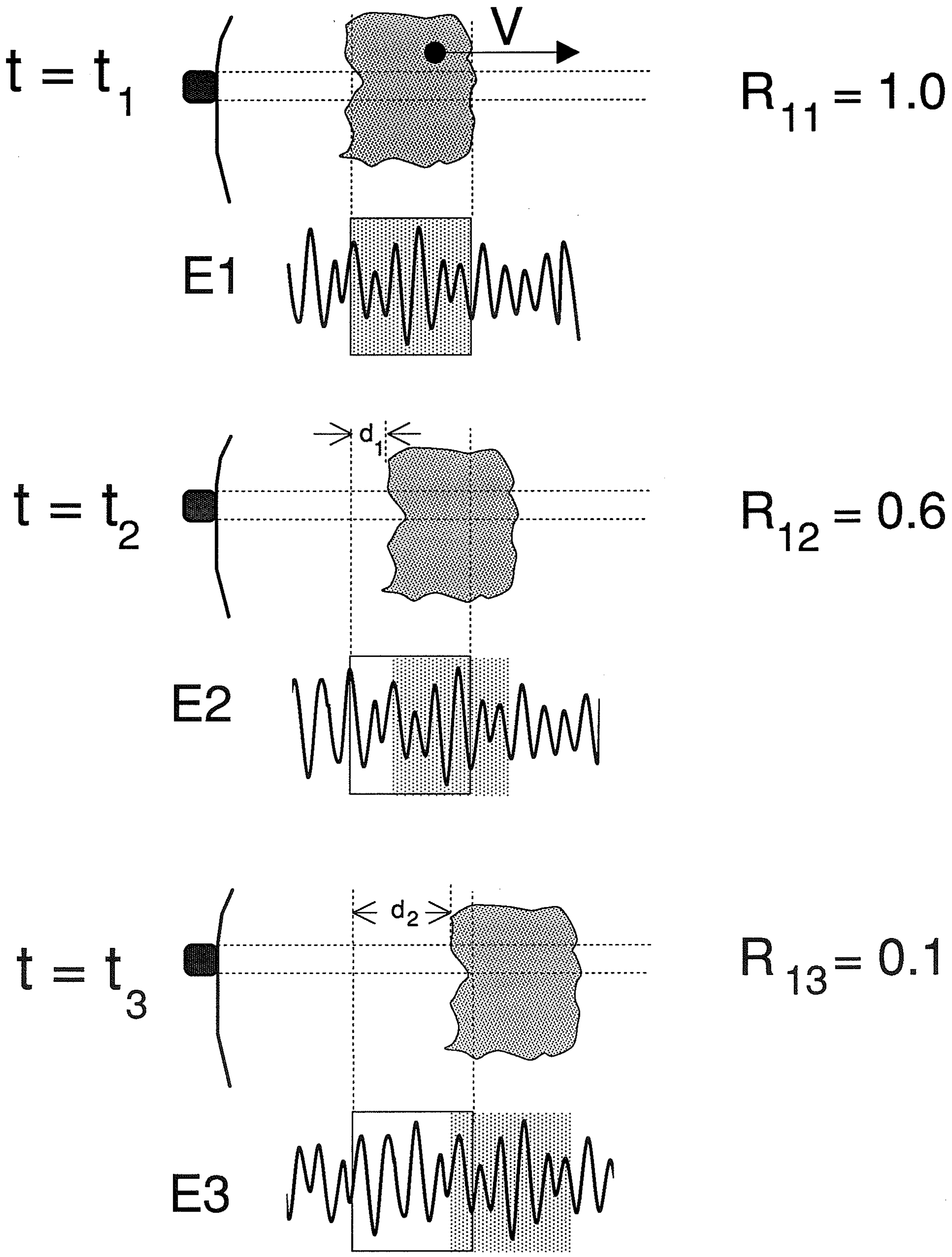


Figure 9 - Variation of the normalized correlation coefficient with tissue motion within the correlation window.

between the 1st and 2nd echo is 0.6. Since the tissue has moved farther between the 1st and 3rd than the 1st and 2nd echoes, $R_{13} = 0.1$ and is smaller. If all of the original tissue moves out of the window between the 1st and nth echo, then $R_{1n} = 0.0$. Thus the change in R between echoes is proportional to the distance the tissue has moved. Note that the value of the correlation coefficient will change not only for axial motion but for lateral and transverse motions as well.

Two of the first researchers to use this method of tissue motion assessment were Dickinson and Hill in 1982 [18]. Figure 10a) shows how the correlation coefficient changes for lateral and axial motions as measured in post-mortem liver tissue. Dickinson and Hill define a modified correlation coefficient $R'=1-R$ where $R'=0$ for no tissue motion. Figure 10b) shows how 1-R changes for oscillatory motion of post-mortem liver tissue.

More recently, Tristam, Barbosa, Cosgrove, Bamber, and Hill [19] have applied this technique to characterize the kinetic response of healthy and diseased liver tissues in human patients. Tristam et al. have digitized ultrasound echoes from the liver and have plotted R' as it varied with the cardiac cycle. Figure 11 shows how R' varies with the cardiac cycle for two patients X and Y with secondary liver deposits. The top two plots show the variation in R' for diseased sections of the liver, and the bottom plots show how R' varies for the healthy sections. The healthy sections have a much larger range of R' than the diseased areas.

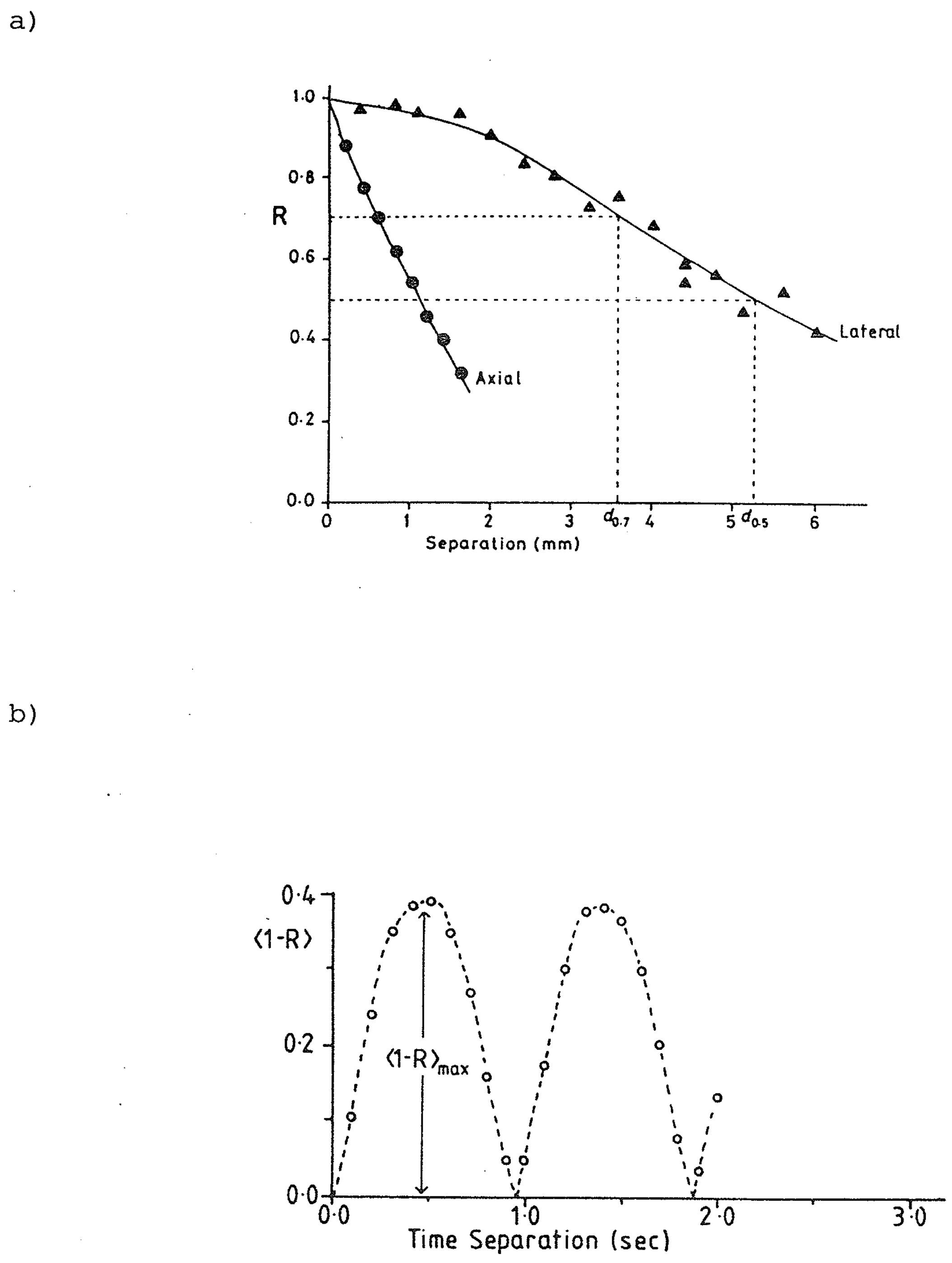


Figure 10 - a) Graph showing the correlation function between two A-scans as a function of separation for lateral and axial motions. b) Graph showing 1-R as a function of time for oscillatory motion of fixed tissue parallel to axis of transducer. (From Dickinson and Hill, [18])

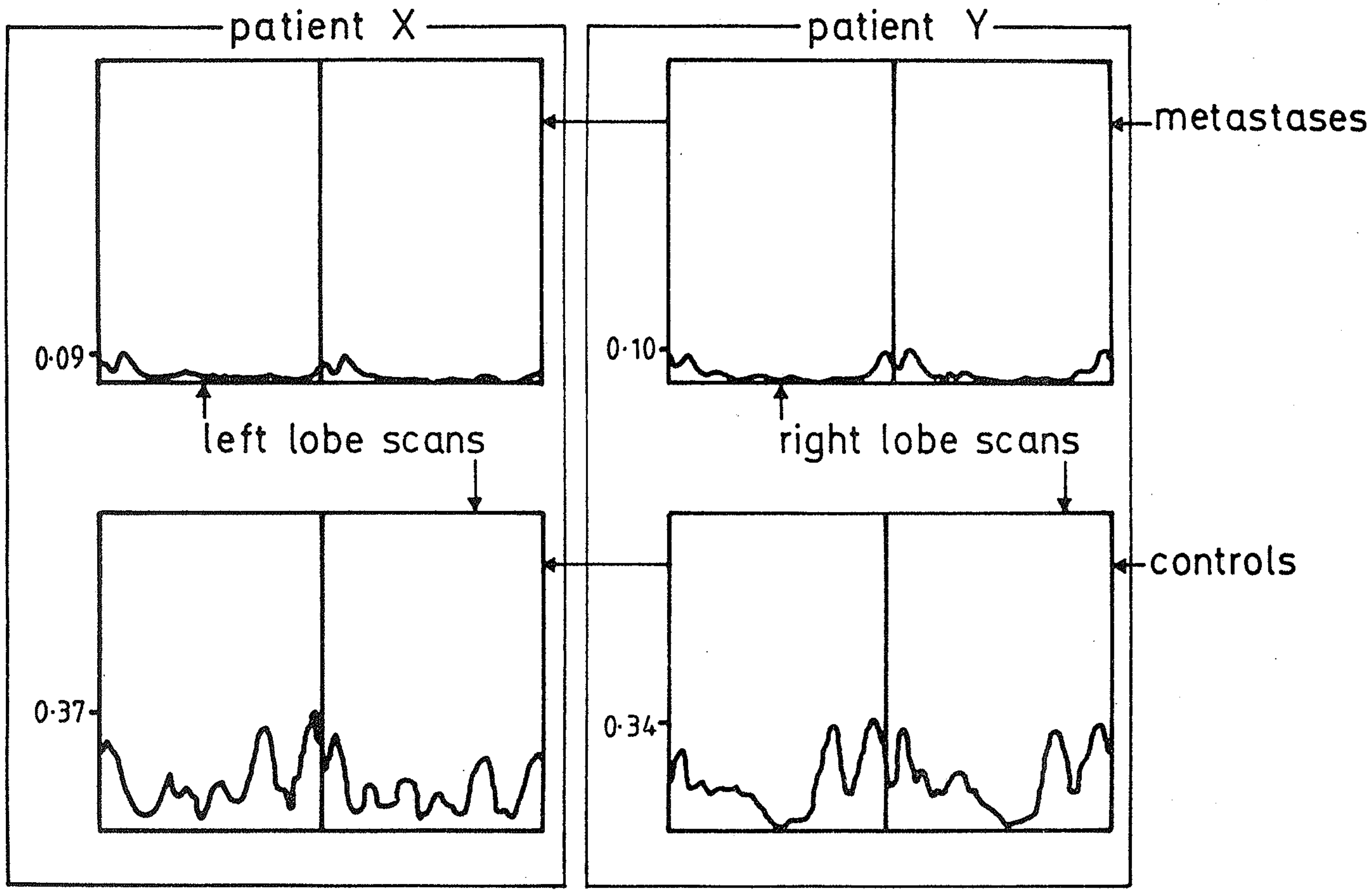


Figure 11 - Èxamples of correlation patterns in two patients X and Y with secondary liver deposits (horizontal axis: time normalized to two cardiac cycles, vertical axis: values of R' = 1-R). (From Tristam, Barbosa, Cosgrove, Bamber, and Hill [19])

Thus it may be possible to use this technique to diagnose healthy and diseased sections of various tissues.

2.2.3.2 Velocity measurement by locating the maximum in the normalized correlation coefficient

The previous correlation method provides a means of determining if there is tissue motion present, but it is not very quantitative. It does not provide information about the direction of motion, and once the original volume has moved out of the correlation window, no more information can be obtained. A more quantitative means of measuring tissue motion is to use the correlation coefficient to track the same group of scatterers. This can be done by varying the window location within the received echoes in order to find the location which produces a maximum in the correlation coefficient. This is illustrated in Figure 12. Suppose that an ultrasonic pulse is transmitted at time $t=t_1$ and an echo E_1 is recorded. If a tissue scatterer group is located at d_1, it will produce an ultrasonic footprint located at a digital distance r of digital length N in E_1. If another pulse is transmitted at $t=t_2$, the scatterers will have moved some unknown distance Δd (shown moved towards the transducer in this case). Its footprint will also have moved accordingly to a new spot in E_2. The distance (in time) the scatterers have moved can be determined by correlating the original footprint in E_1 with different equivalent length sections of E_2. Mathematically, the process can be thought of as shifting the windowed footprint of

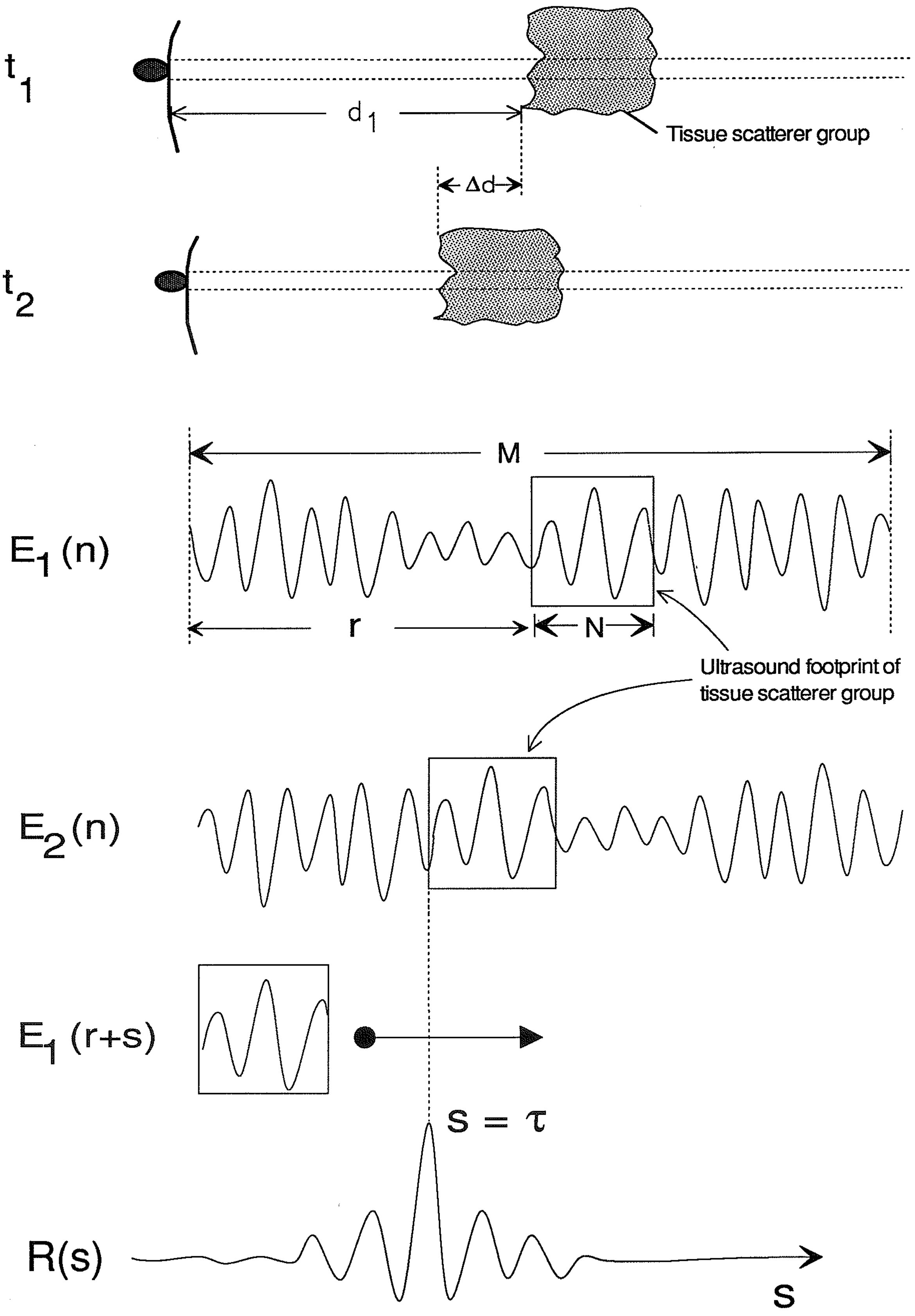


Figure 12 - Tracking a tissue volume by finding where the correlation coefficient becomes maximum.

E_1 by s and varying s until the same footprint is found in E_2. The correlation coefficient R(s) is calculated for each value of s, and the value of $s=\tau$ producing the maximum in R(s) corresponds to the new location of the tissue scatterers. The distance and the velocity of the tissue scatterers can then be calculated if the speed of sound in the tissue is known.

This method of tracking a specific group of scatterers is under development by a number of researchers in the application of measurement of blood flow. If $s=\tau$ is the time shift producing a maximum in R(s), then the axial velocity of the tissue scatterers (in this case a volume of blood cells) can be calculated by

$$v_a = \tau c/(2T) \tag{9}$$

where c is the speed of sound and T is the period between transmission of the two ultrasonic pulses.

The use of time-domain correlation to measure blood flow in real time is the main topic of this dissertation, and the details of this technique as developed at the Bioacoustics Research Laboratory at the University of Illinois are explained in Chapter 3.

2.2.3.3 Estimating disturbedness of flow from the value of the correlation coefficient

Theoretically, the value of the correlation coefficient at its maximum will be one if the two windowed sections of echo overlap exactly and if the footprint stays exactly the same. In reality, there are a number of factors which can change the shape of the footprint and degrade the value of the correlation coefficient from one at its maximum. Bonnefous [20] has analyzed two causes of this degradation: velocity dispersion within the range cell and the presence of transverse flow velocity components. Velocity dispersion relates to the fact that the correlation window in the echo corresponds physically to a volume in the blood vessel. The size of the volume depends on the transducer beamwidth and the length N of the correlation window (which will have a minimum length in order to produce a meaningful correlation). In many cases of blood flow, the velocity in this minimum volume will not be constant but will have a distribution of velocities. In addition, there may be transverse flow components present which will have no axial flow component. Bonnefous has estimated the value of the correlation coefficient at its maximum as [20]

$$R(\tau) \approx (1-Q)(1-2\pi^2 v_0^2 \sigma^2[\tau]) \qquad (10)$$

where Q is a measure of the transverse flow velocity components and the $2\pi^2 v_0^2 \sigma^2[\tau]$ term is a measure of the velocity dispersion within the range cell (v_o is the mean velocity and $\sigma[\tau]$ is the variance of the velocity at the maximum correlation in the range cell). Both of these terms serve to reduce the correlation coefficient from its theoretical maximum value of one. Hence, the value of the correlation coefficient at its maximum can be used as an indicator of the disturbedness of the blood flow. Bonnefous has developed a system which can plot the mean velocity and velocity variance of blood flow, and has used this to diagnose cardiovascular disorders such as stenosis. Figure 13a) shows the velocity and variance profiles in a healthy carotid and Figure 13b) the profiles in a carotid with severe stenosis, where the brightness represents a higher value of velocity or variance. The variance in the healthy individual is fairly small; however, it is much higher in the carotid with the severe stenosis. Thus, this technique may be useful in diagnosing cardiovascular disorders which are characterized by disturbed flow.

2.2.3.4 Maximum likelihood estimation

Ferrara and Algazi [21] have developed correlation techniques using the maximum likelihood estimation point of view. This method utilizes both the shift in time and shift in frequency produced by moving scatterers. They define a wideband point maximum likelihood estimator (MLE) where the likelihood of a velocity v_i is given as

$$l(v_i) = \left| \sum_{k} \int_{-\infty}^{\infty} r'(t)s'^{*}(t-d-kT[1+2v_i/c]) \exp[-j\sigma v_i t]\, dt \right|^2 \qquad (11)$$

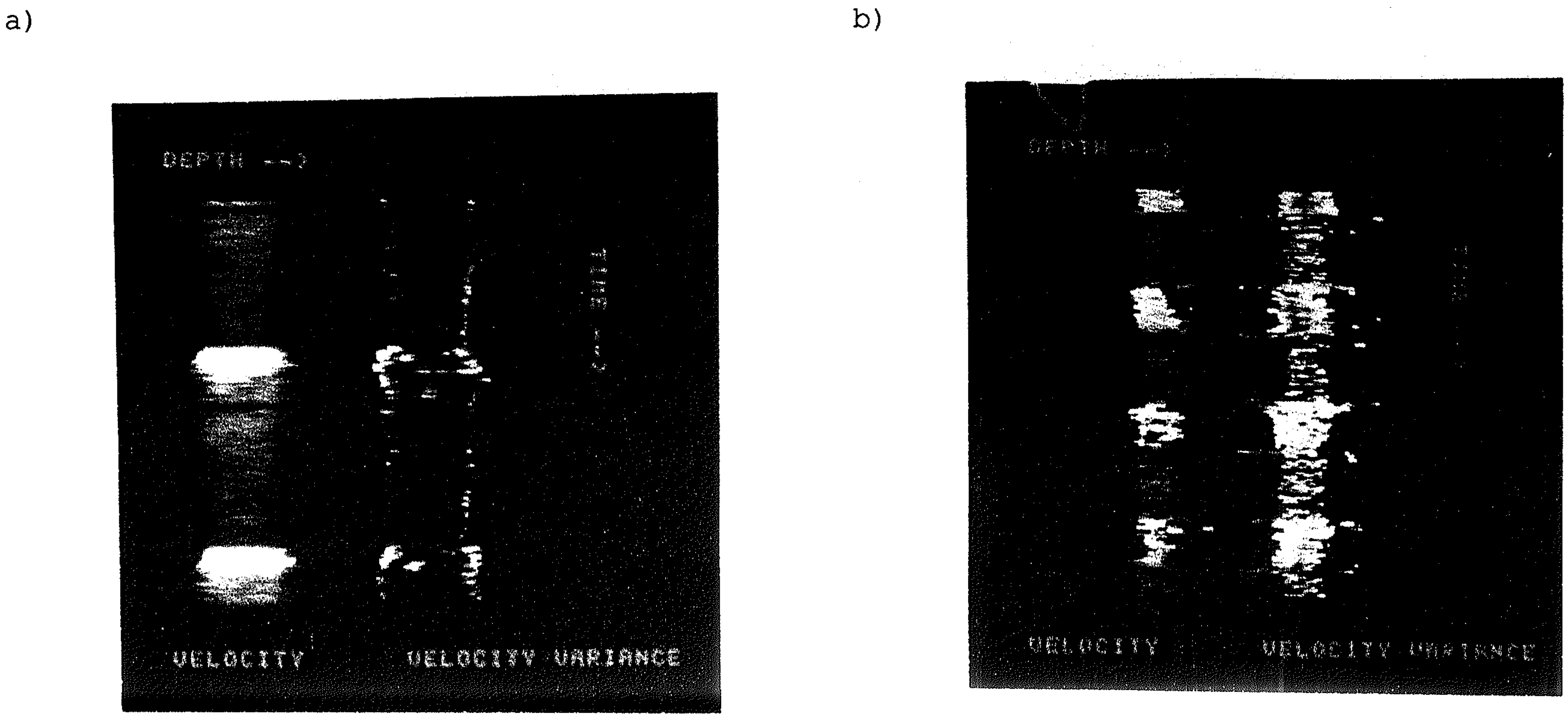


Figure 13 - Velocity and variance profiles in a) healthy carotid and b) carotid with a severe stenosis. (From Bonnefous, [20])

where $l(v_i)$ is the likelihood of a velocity v_i, $r'(t)$ is the complex envelope of the received signal, $s'(t-d)$ is a delayed version of the transmitted signal (which includes the effects of signal propagation and scattering), and $\sigma = 2\omega_0/c$ where ω_0 is the transducer center frequency and c is the speed of sound in the medium. The mean velocity can be calculated from the ratio of the first moment of the likelihood function to the zeroth moment:

$$v_m = \frac{\sum_i v_i l(v_i)}{\sum_i l(v_i)} \qquad (12)$$

where v_m is the mean velocity. The output of the wideband point MLE is likelihood vs. velocity. This is shown in Figure 14a) for the velocity measured near the center of a 7 mm vessel which has a fully developed laminar flow. The shape of the likelihood peak will depend on the velocity dispersion within the measurement cell. If the velocities in the cell are nearly uniform, the likelihood peak will be narrow and tall. If there is significant velocity dispersion within the cell, then the peak will be wider and flatter. Ferrari and Algazi have plotted the likelihood vs. position at different points across the 7 mm vessel in a three-dimensional format, as shown in Figure 14b). This plot shows that the velocity is maximum and has the least amount of dispersion at the center of the vessel (near position 625). The velocity

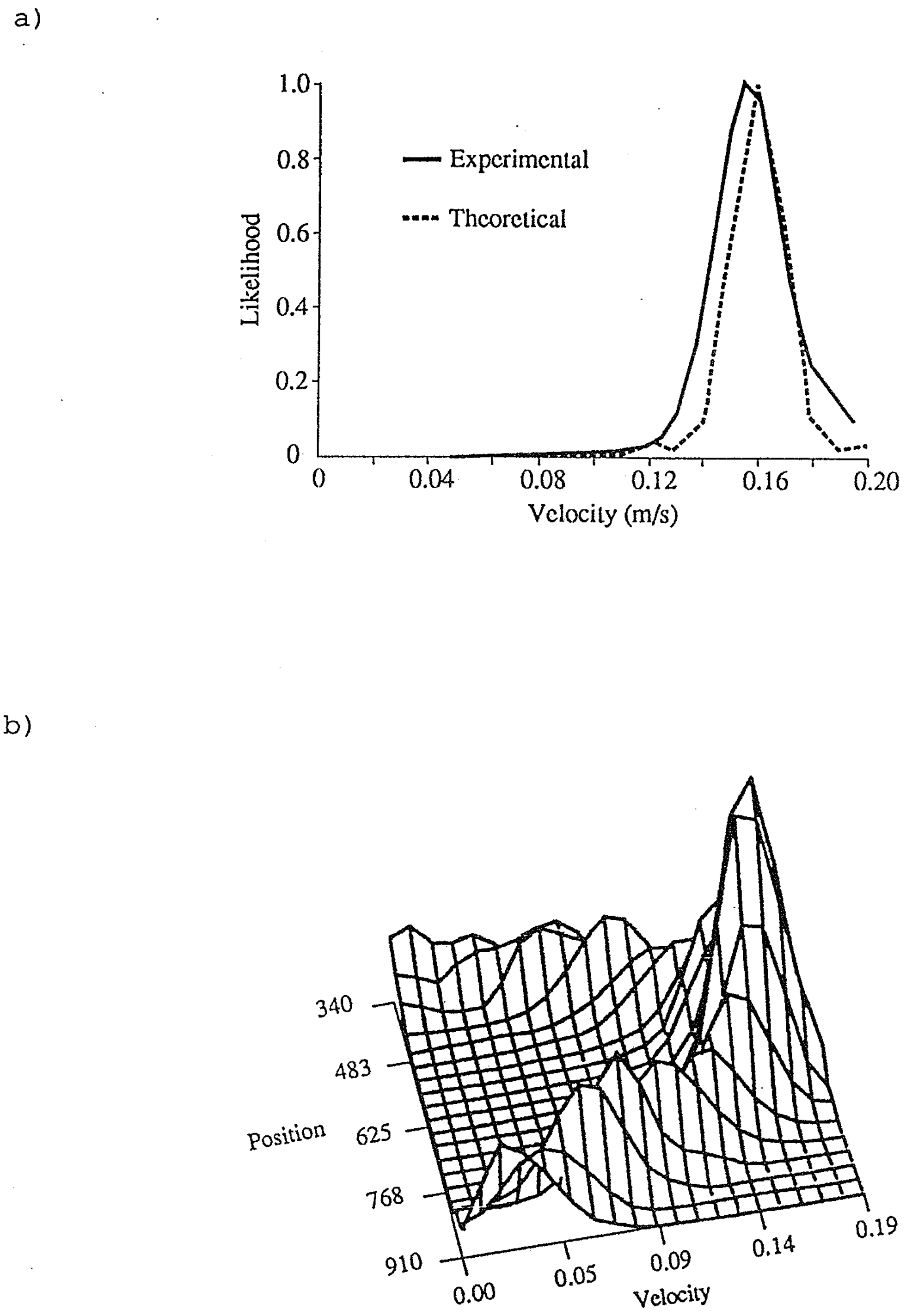


Figure 14 - Experimental and theoretical resolutions of the wideband point MLE for the velocity near the center of a 7 mm vessel. b) Three-dimensional plot of velocity likelihood at different points across the vessel. (From Ferrara and Algazi, [21])

decreases and the dispersion increases as the vessel walls are approached (near positions 340 and 910). This is characteristic of fully developed laminar flow within a vessel. Thus this technique can also be used to estimate the amount of disturbedness of flow within a blood vessel.

2.3 Two-Dimensional Measurement Configurations

The two-dimensional tissue motion assessment techniques described in this section refer to measurements made by the frame-to-frame analysis of ultrasound speckle patterns in B-mode images. This configuration is shown in Figure 15, where moving tissue will produce a changing speckle pattern in the ultrasound image. In general, a volume of tissue which is in motion can undergo a number of possible changes. It can retain its shape and simply be translated or rotated. In more complex motion, its shape can change (deformation), or it can change density (be compressed or expanded). The change in the tissue will produce a corresponding change in the speckle pattern observed on the ultrasound image. The amount of change in the speckle pattern will in part depend on the difference in the number of frames over which the comparisons are made. In some applications, such as the measurement of blood flow, measurements are made over consecutive frames. In other applications, such as assessing tissue contractility, comparisons may be made over a wider difference in frames.

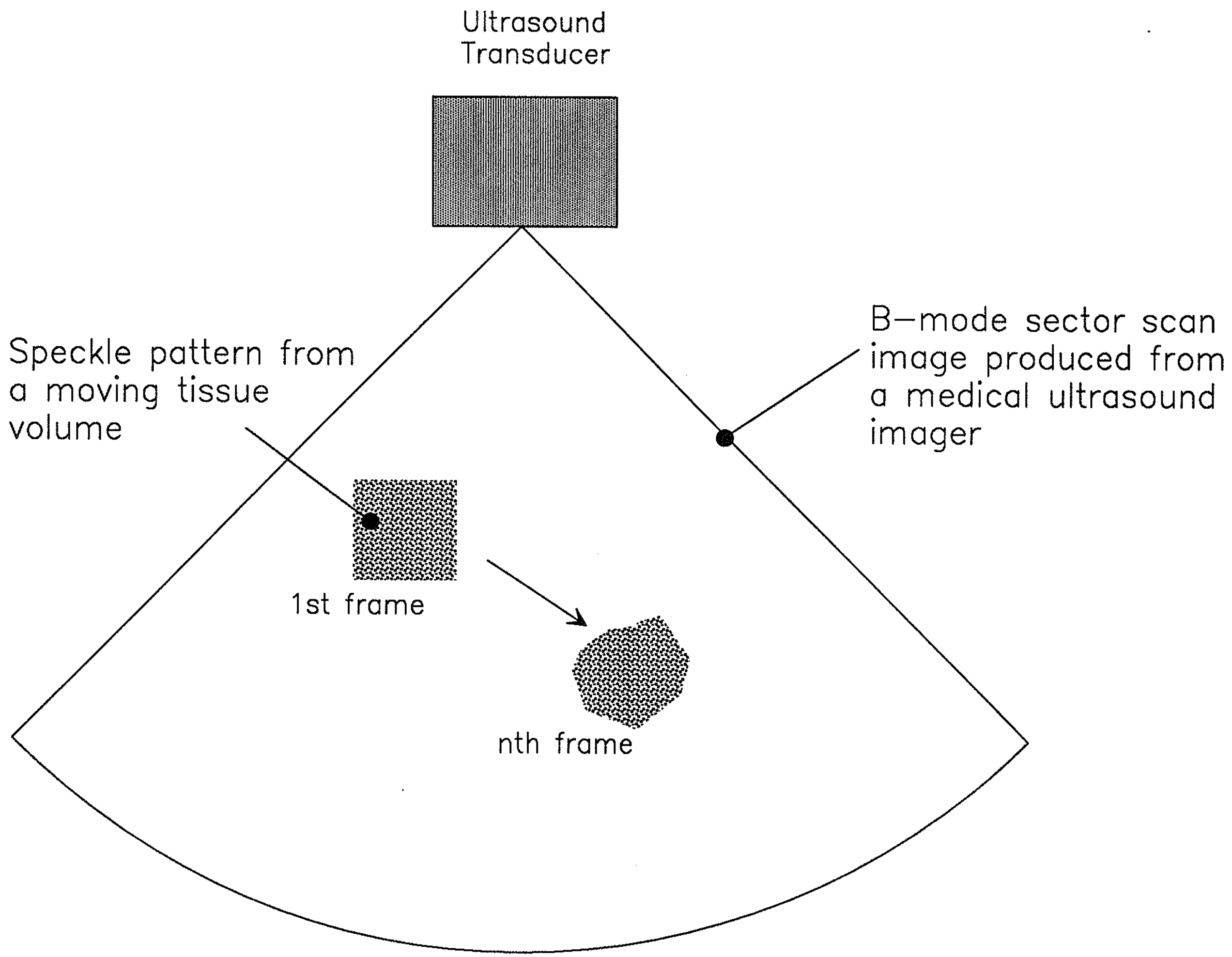

Ultrasound Transducer
Speckle pattern from a moving tissue volume
B–mode sector scan image produced from a medical ultrasound imager
1st frame
nth frame

Figure 15 - Tissue motion detection by frame to frame analysis of ultrasound speckle pattern in B-mode images. The tissue can undergo translation, rotation, deformation, or contraction between different image frames.

The two-dimensional tissue assessment methods have been divided into two main categories: optical flow algorithms and correlation-type algorithms.

2.3.1 Optical flow techniques

Optical flow methods calculate the velocity vector of flow in an image by deriving an equation that relates the change in image brightness at a particular point to the motion of the brightness pattern [22]. The optical flow can be denoted by

$$\vec{V} = \{\ U(x,y),\ V(x,y)\ \} \tag{13}$$

where U and V are the velocity vector components at the point (x,y) in the image. The optical flow algorithm assumes that if a point in the image moves by δx in the x direction, δy in the y direction, and δt in time, the brightness E of that point remains constant:

$$E(x,y,t) = E(x+\delta x,\ y+\delta y,\ t+\delta t) \tag{14}$$

This leads to what is referred to as the brightness constraint:

$$E_x U + E_y V + E_t = 0 \tag{15}$$

where E_x, E_y, and E_t are the partial derivatives with respect to x, y and t. These partial derivatives can be estimated from the image, as shown in Figure 16. Each partial derivative is estimated by taking the average of four differences taken over adjacent measurements in a cube. The solution of $E_xU+E_yV+E_t=0$ will not, however, determine the component of motion in the direction of constant brightness contours. An additional constraint is needed which assumes that the velocity field of the brightness patterns in the image varies smoothly almost everywhere. The measure of the departure from smoothness in the velocity flow can be written as

$$E_c^2 = \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \tag{16}$$

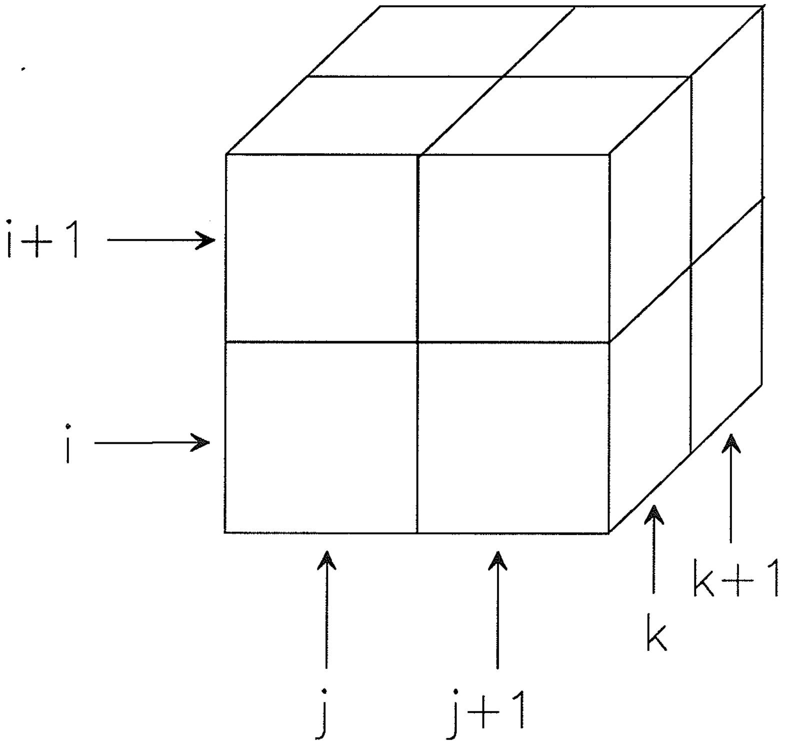
The error in the rate of change of image brightness can be expressed as

$$E_b = E_xU + E_yV + E_t \tag{17}$$

and the total error to be minimized is

$$E^2 = \iint (\alpha^2 E_c^2 + E_b^2)\ dxdy \tag{18}$$

where α is a weighting parameter set roughly equal to the noise in the estimate of E_x+E_y. The velocity vector can be solved with the minimization of Equation (18) and the estimated partial derivatives using various iterative methods such as the Gauss-Seidel method [22].



$$
\begin{aligned}
E_x = [\ & E(i,\ j+1,\ k) - E(i,\ j,\ k) + E(i+1,\ j+1,\ k) \\
& -E(i+1,\ j,\ k) + E(i,\ j+1,\ k+1) \\
& -E(i,\ j,\ k+1) + E(i+1,\ j+1,\ k+1) \\
& -E(i+1,\ j,\ k+1)\ ]/4
\end{aligned}
$$

$$
\begin{aligned}
E_y = [\ & E(i+1,\ j,\ k) - E(i,\ j,\ k) + E(i+1,\ j+1,\ k) \\
& -E(i,\ j+1,\ k) + E(i+1,\ j,\ k+1) \\
& -E(i,\ j,\ k+1) + E(i+1,\ j+1,\ k+1) \\
& -E(i,\ j+1,\ k+1)]/4
\end{aligned}
$$

$$
\begin{aligned}
E_t = [\ & E(i,\ j,\ k+1) - E(i,\ j,\ k) + E(i+1,\ j,\ k+1) \\
& -E(i+1,\ j,\ k) + E(i,\ j+1,\ k+1) \\
& -E(i,\ j+1,\ k) + E(i+1,\ j+1,\ k+1) \\
& -E(i+1,\ j+1,\ k)]/4
\end{aligned}
$$

Figure 16 - The three partial derivatives of image brightness at the center of a cube are estimated from the average of first differences along 4 parallel edges of a cube. Here j corresponds to the x direction, i to y, and k in the time direction. (From Horn and Schunck, [22])

Once the velocity vector $\vec{V}(x,y)$ has been determined for each pixel location, the direction and magnitude of the flow can be plotted at each pixel location. This is illustrated in Figure 17 for simple rotation and simple contraction of a brightness pattern [22]. Each vector consists of a direction and a magnitude which describes the flow velocity at that pixel.

2.3.1.1 Optical flow applied to echocardiography

One application of the optical flow method is in the field of echocardiography, which is widely used in clinical cardiology but is subject to a large observer variability. Optical flow techniques have the potential to more quantitatively assess the condition of the heart. Mailloux, Langlois, Simard, and Bertrand [23] have utilized this optical flow method to compute the interframe optical flow velocity field of the heart, which is illustrated in Figure 18. This figure shows the direction of motion of local areas of the heart muscle during contraction.

The use of the optical flow method in this configuration describes the total motion of the heart. It may be clinically useful to know what components of the overall motion are due to translation, rotation, deformation, and contraction of the cardiac muscle. The addition of the constraint of local linearity to the optical flow algorithm will allow these individual components to be extracted. Any vector field can be considered locally linear, describable by linear equations of the form

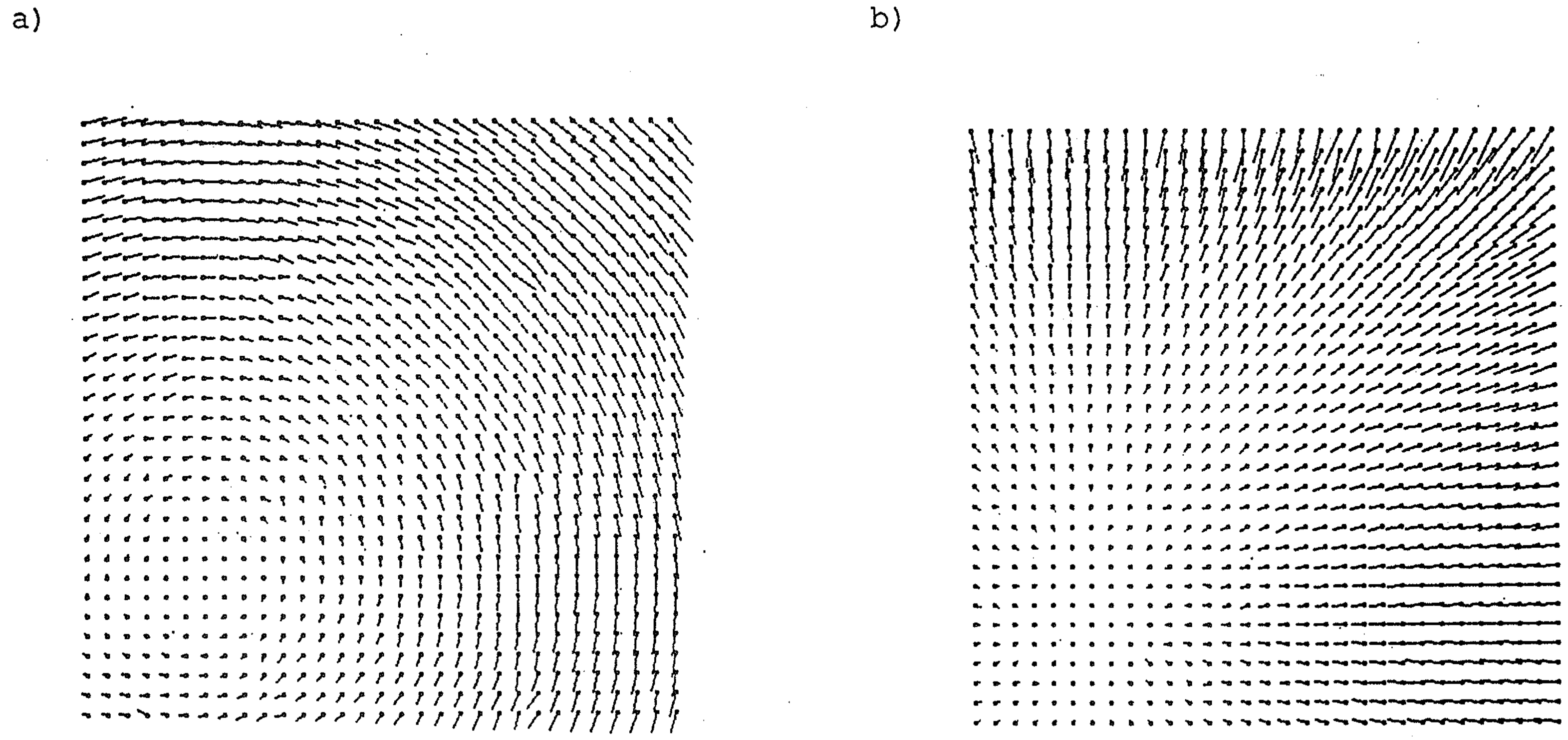


Figure 17 - Optical flow patterns for a) simple rotation and b) simple contraction of a brightness pattern. (From Horn and Schunk, [22])

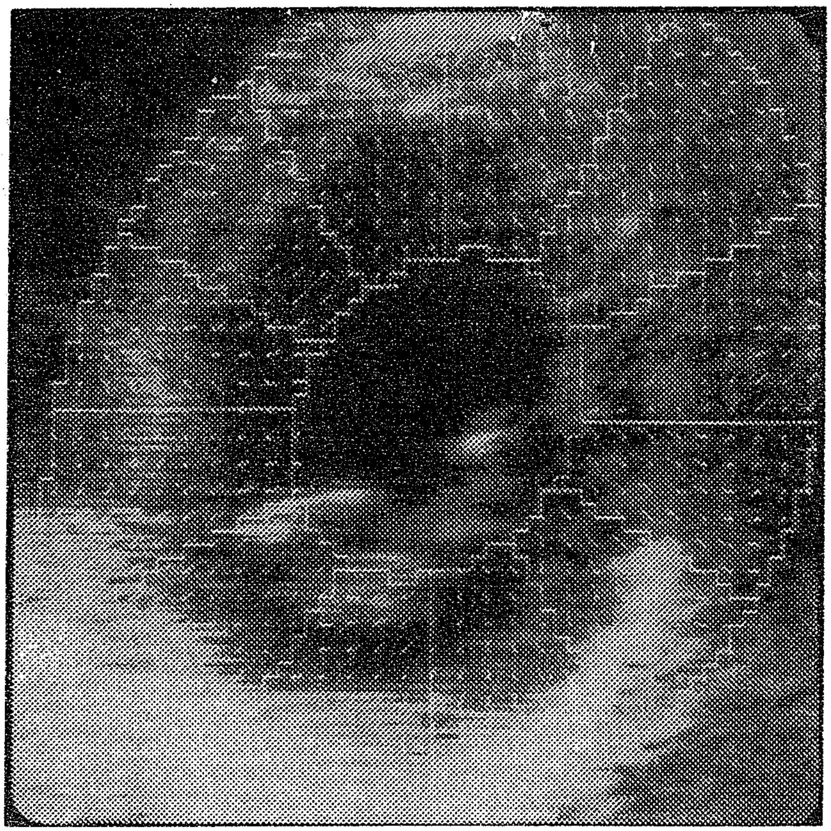


Figure 18 - Interframe optical flow velocity field of the heart. (from Mailloux, Langlois, Simard, and Bertrand, [23])

$$\begin{bmatrix} U(x,y) \\ V(x,y) \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} + \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{19}$$

These can be rewritten as

$$\begin{aligned} \begin{bmatrix} U(x,y) \\ V(x,y) \end{bmatrix} &= \begin{bmatrix} a \\ b \end{bmatrix} && \text{translational} \\ &+ (A + D)/2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} && \text{divergent} \\ &+ (C - B)/2 \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} && \text{rotational} \\ &+ \begin{bmatrix} (A - D)/2 & (B + C)/2 \\ (B + C)/2 & (D - A)/2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} && \text{shear} \end{aligned} \tag{20}$$

where the right-hand terms of this equation describe translational, divergent, rotational, and shear fields [23]. The components a, b, A, B, C, and D can be obtained after computing the OF and then minimizing

$$\sum_{n} [(a+Ax_n+By_n-U_n)^2 + (b+Cx_n+Dy_n-V_n)^2] \tag{21}$$

by the method of least squares where the summation is taken over all points in the image where the velocity is observed. Mailloux et al. [23] have used this method to break up the interframe optical flow image into these separate components. A major disadvantage of this method is the extreme computation time. The system described by Mailloux et al. requires approximately 4 hours to process the images [24].

2.3.1.2 Optical flow applied to assessment of contractility of skeletal muscle

A somewhat simpler method of calculating the optical flow is described by Bertrand, Meunier, Doucet, and Ferland [25]. Their method uses only the constraints of constant brightness (Equation (15)) and local linearity (Equation (19)). Combining these two constraints yields

$$E_xa + E_xAx + E_xBy + E_yb + E_yCx + E_yDy = -E_t \tag{22}$$

which can be solved for a, b, A, B, C, D using a least squares fit. A set of matrices describing the motion of the tissue between the 0th and the kth frame can be written as

$$\begin{bmatrix} x_k \\ y_k \end{bmatrix} = T_{ko} + M_{ko} \begin{bmatrix} x_o \\ y_o \end{bmatrix} \tag{23}$$

where T_{ko} is a translational matrix and M_{ko} is the product of a rotational matrix R_{ko} and a biaxial deformation matrix D_{ko} [25] where

$$R_{ko} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{24}$$

and

$$D_{ko} = \begin{bmatrix} \alpha & \gamma \\ \gamma & \beta \end{bmatrix} \tag{25}$$

The eigenvalues λ_1 and λ_2 of the Hermitian deformation matrix D_{ko} represent deformation along the principal axes, with their associated eigenvectors giving the orientation of the strain. If the material under the linear transformation is incompressible, then

$$\lambda_1\lambda_2\lambda_3 = 1 \tag{26}$$

where λ_3 is the deformation perpendicular to the image plane.

Bertrand et al. [25] have used this method to assess the contractility of skeletal muscle in humans. They have imaged the forearm and assessed the contractility of muscles in the forearm from the speckle patterns in the B-mode image. Figure 19a) shows the structure of the muscles in the forearm. The rectangular area indicates the region of interest. The muscle T is used in the movement of the third finger and muscle L is used to move the little finger. Contractility maps of the λ_3 parameter are shown in Figures 19b) and 19c) for movement of the third finger and movement of the little finger, respectively. Figure 19b) indicates contraction in the area of muscle T, and Figure 19c) indicates contraction of muscle L. The contractility regions are exclusive of each other as they should be. Thus this method can sense muscle contractions and may be useful in assessing the contractility of different types of muscle.

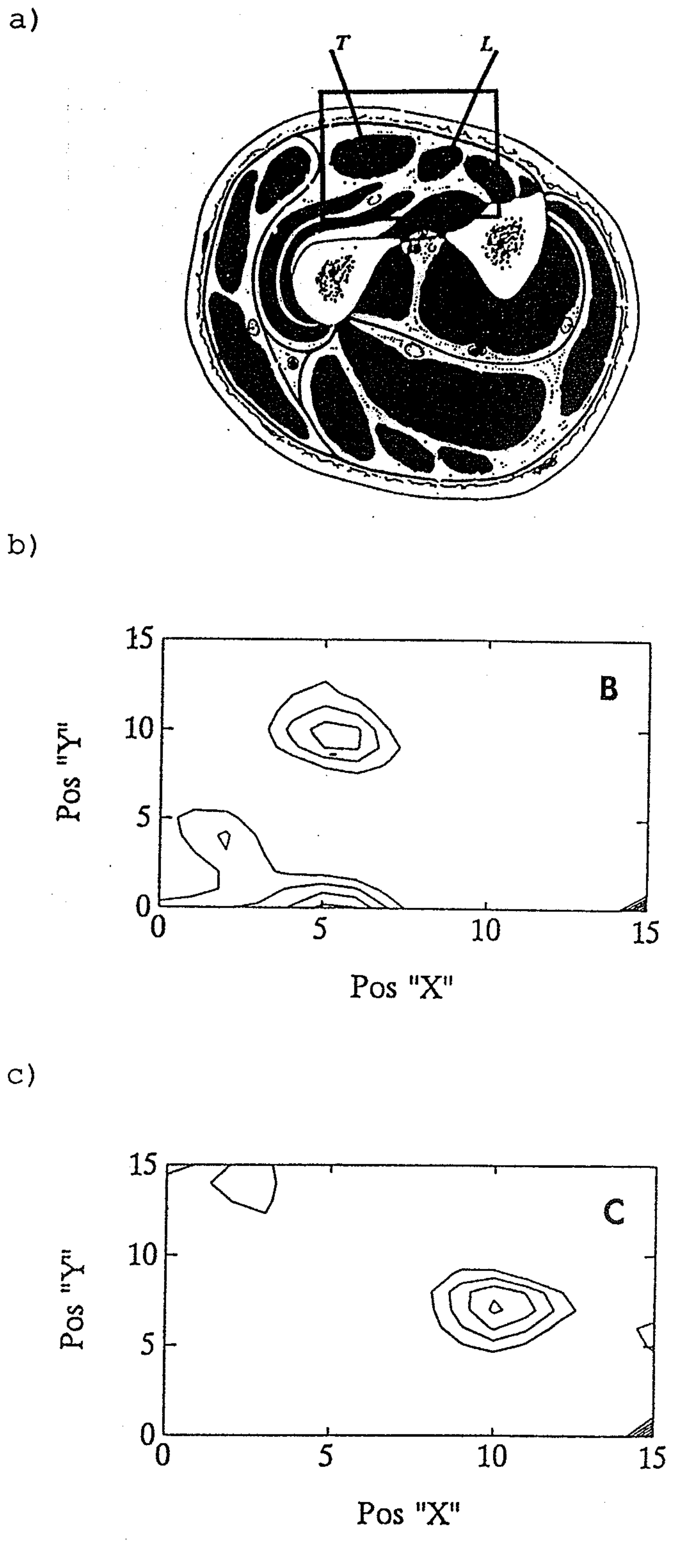


Figure 19 - a) Anatomy of muscle structure in the forearm. b) Isocontractility map at maximum contraction of the third finger. c) Isocontractility map at maximum contraction of the little finger. (From Bertrand, Meunier, Doucet, and Ferland [25])

2.3.2 Measurement of blood flow from B-scan images

The measurement of blood flow from speckle patterns in B-mode images consists of tracking the speckle pattern from a given volume of blood cell scatterers as it moves between frames. The motion of the speckle pattern as it moves in a blood vessel is shown in Figure 20. With a two-dimensional imaging system, it is possible to measure both the lateral $\vec{L}$ and axial $\vec{A}$ components of flow, from which the actual velocity vector $\vec{V}$ can be directly reconstructed. One-dimensional geometries are capable of measuring only the axial component $\vec{A}$.

2.3.2.1 Blood flow detection using Boolean operators

One of the simplest means to detect blood flow (or tissue motion in general) from consecutive B-mode images is to simply subtract one image from another. Pixel locations with no change in brightness will show up with zero brightness in the subtracted image, and those with changes in brightness will show up with the difference in brightness. This method is effective for showing large differences between frames but maps small differences from blood flow into a few gray levels near zero with very little contrast [26]. A better method is to perform bit-by-bit Boolean operations in identically positioned pixels between frame buffers. Gardiner and Fox [26] have used this method with the Boolean operations AND, OR, and XOR to produce *in-vivo* color flow images. Figure 21 shows color flow images using the OR and XOR operations

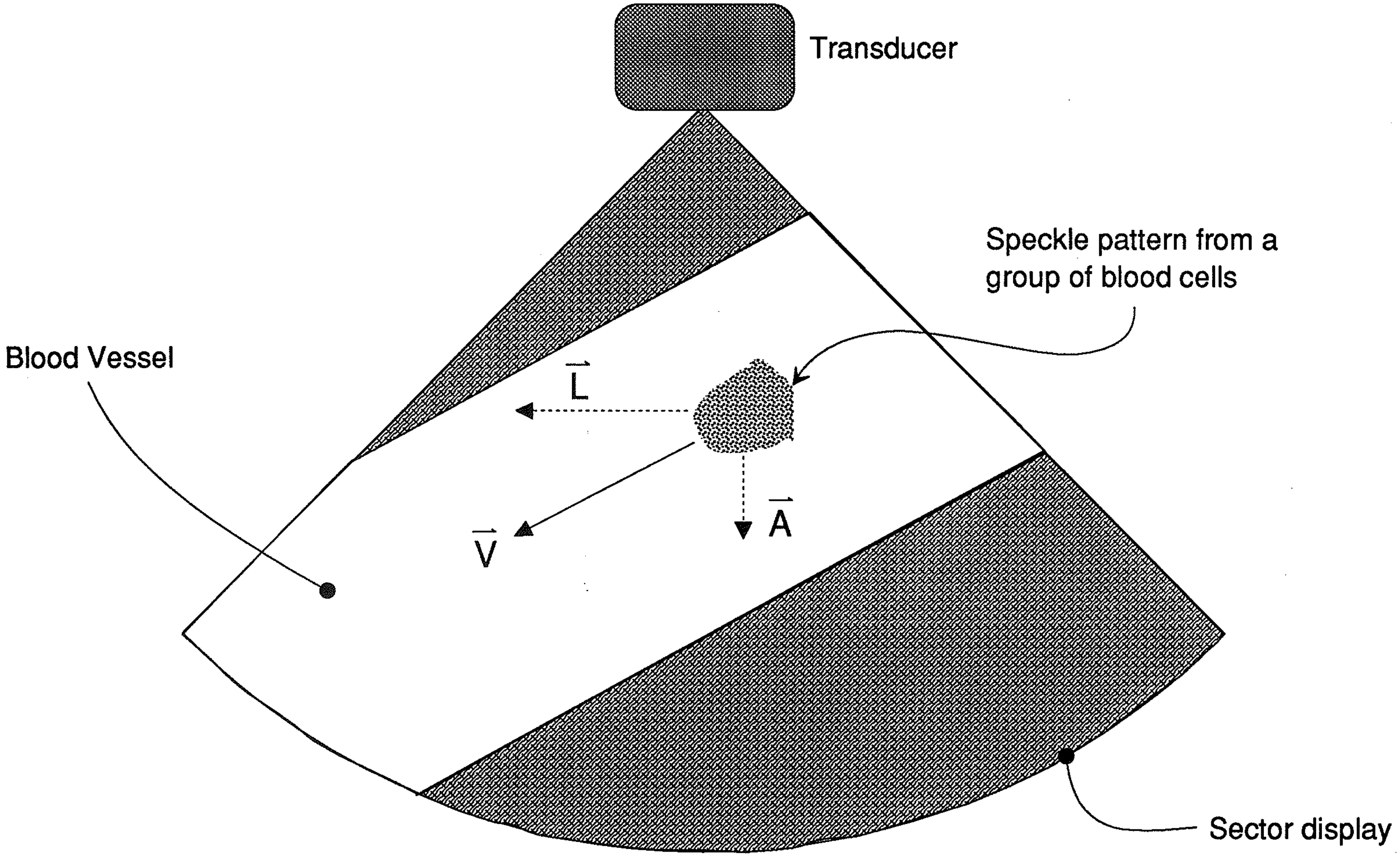


Figure 20 - Blood flow detection by analysis of speckle patterns in B-mode images. Both the lateral and axial components of velocity can be estimated.

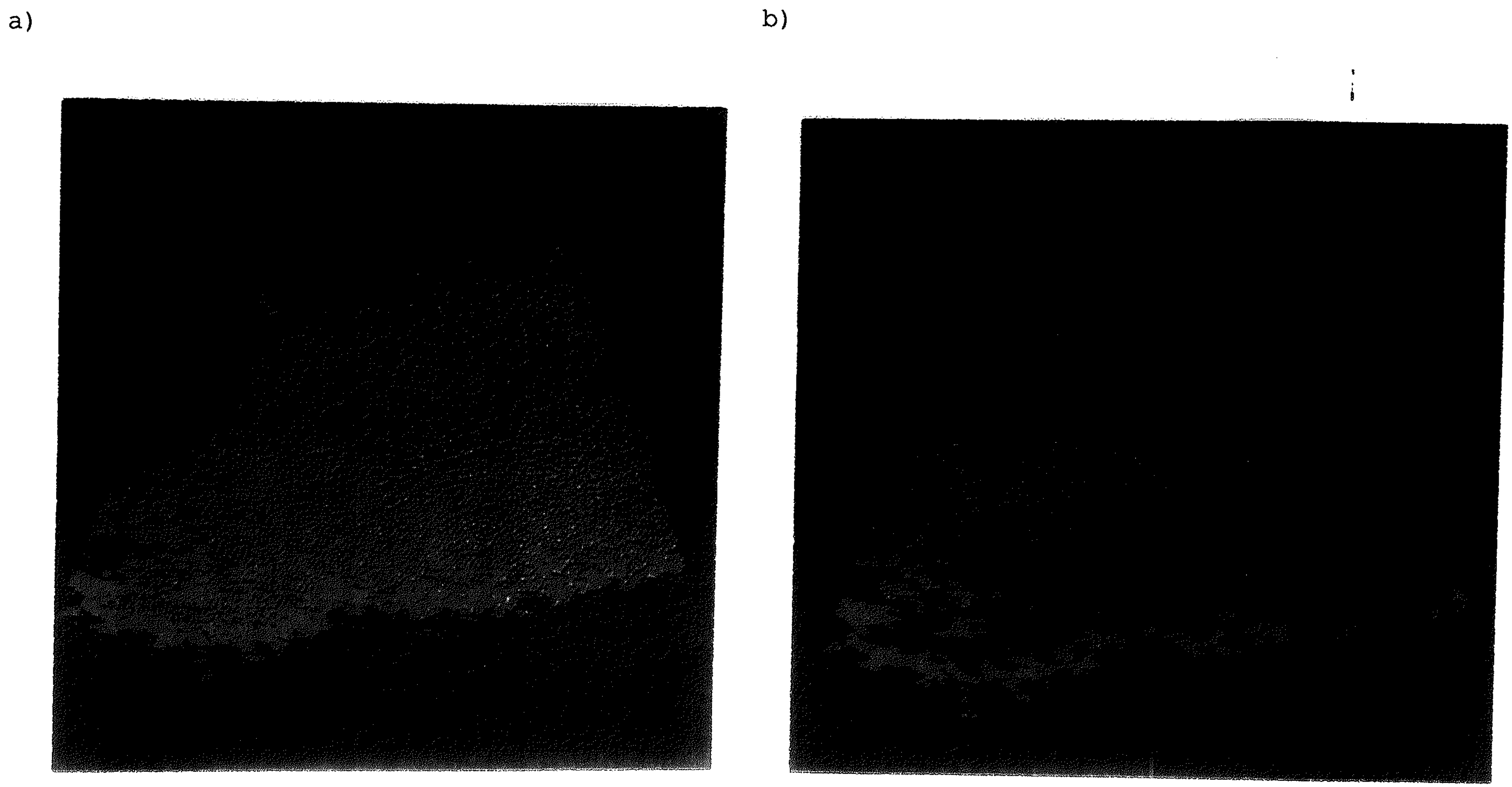


Figure 21 - Color-flow images processed from frames of the a) OR and b) XOR images. Red indicates areas of high-velocity flow. (From Gardiner and Fox, [26])

for flow measured in the abdominal aorta of a human. This technique has the advantages that it can easily be performed in real time, and also does not have the angle dependence that Doppler measurements do. The major disadvantage of this technique is that it is only qualitative; it gives an indication that there is motion, but provides no information about the direction of motion.

2.3.2.2 Two-dimensional correlation search algorithm

In order to obtain the direction of motion of blood cell scatterers, the scatterers must be tracked. The method of tracking scatterers using correlation in one dimension can also be applied to two dimensions. Trahey, Hubbard, and von Romm [27] have described a two-dimensional correlation search algorithm whose geometry is shown in Figure 22. In this figure, a group of blood cell scatterers is in location 1 in the first image frame (call it frame X) and has moved to location 2 in the second image frame (frame Y). The scatterers can be tracked by taking a target region from the first frame and sweeping over a search region from the second frame to find out where in the search region the original target region has moved to. This is done mathematically by calculating a normalized two-dimensional correlation coefficient at each location of the target region in the search region:

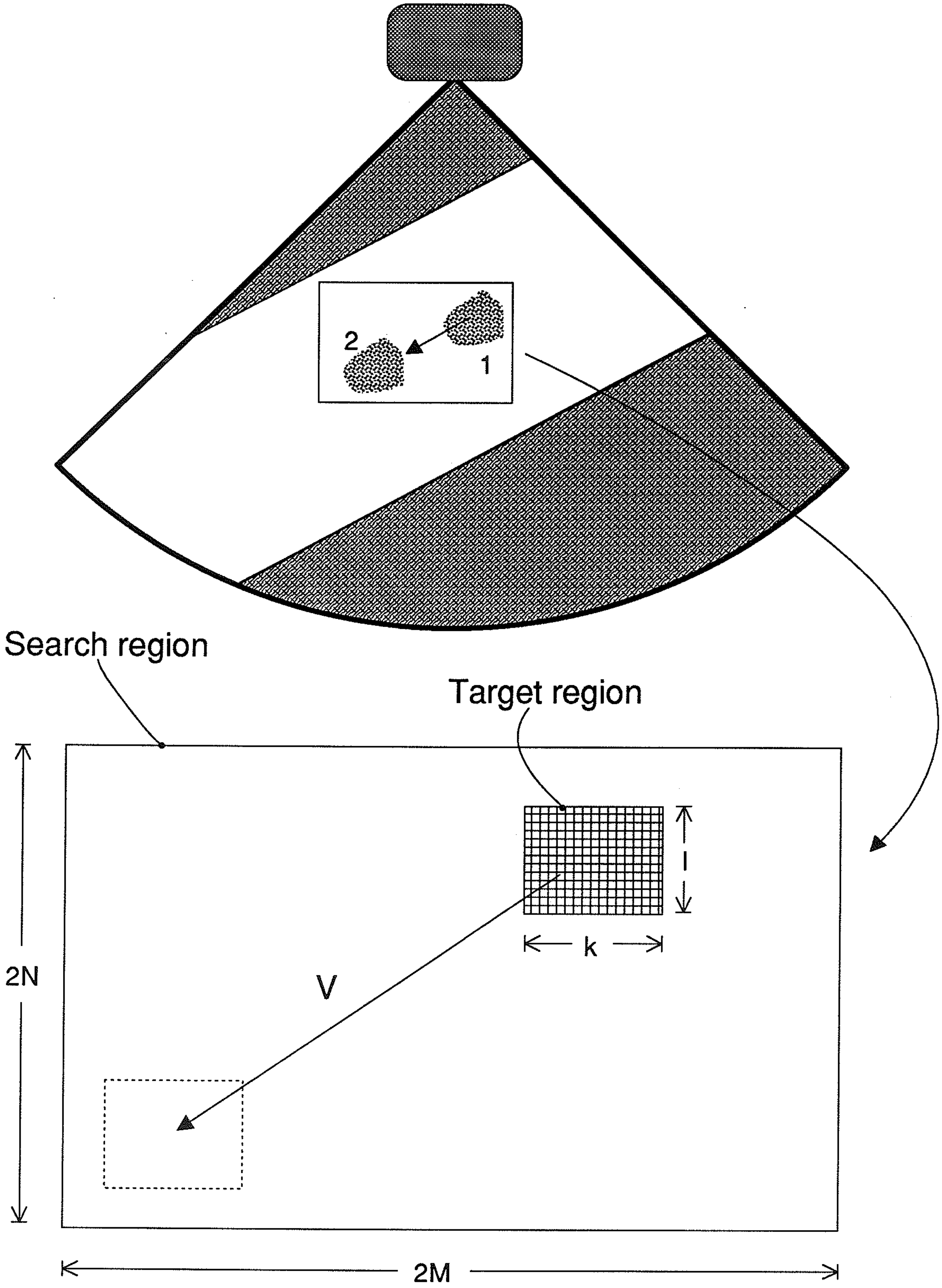


Figure 22 - Two-dimensional correlation search algorithm geometry. The target region is swept across the search region and the 2-D correlation coefficient is calculated at each position in the search region. (from Trahey, Hubbard, and von Romm, [27])

$$\rho_{m,n} = \frac{\sum_{i=1}^{1}\sum_{j=1}^{k}(X_{i,j}-\bar{X})(Y_{i+m,j+n}-\bar{Y})}{\sqrt{\sum_{i=1}^{1}\sum_{j=1}^{k}(X_{i,j}-\bar{X})^{2}(Y_{i+m,j+n}-\bar{Y})^{2}}} \qquad (27)$$

where $\rho_{m,n}$ is the value of the normalized correlation coefficient between a 1 x k region in frame X and the corresponding region displaced by m,n in frame Y, $X_{\_,\_}$ and $Y_{\_,\_}$ are pixel locations in the corresponding images, and $\bar{X}$ and $\bar{Y}$ are the mean pixel values of the corresponding regions. The location m,n which produces the maximum in $\rho_{m,n}$ is considered to be the new location of the scatterers. The scatterer velocity can then be calculated if the frame rate is known. This normalized 2-D correlation coefficient is very similar to the 1-D correlation coefficient given in Equation (8) except that it is carried out over two dimensions. This method also has the advantage that it does not have the angle dependence problem that Doppler does.

A disadvantage of the 2-D method is that the lateral and axial resolutions will not be the same. The axial resolution will be limited only by the transducer frequency, while the lateral resolution is limited by the spacing of the A-scans that form the ultrasound image. For typical imaging systems, the lateral resolution will be several times coarser than the axial resolution [27]. In addition, if the system produces a sector image, the lateral resolution will become coarser at farther distances from the transducer since the pixel spacing will become wider. A 2-D system will also require a longer data acquisition time. For a 1-D measurement system, only single A-lines need to be captured. For a 2-D system, entire frames need to be captured. This may limit the maximum measurable velocity, which will be dependent on the frame acquisition rate.

2.3.2.3 Sum-absolute-difference tracking algorithm

The calculation of the 2-D normalized correlation coefficient can be very math intensive for large search regions. Another method that can be used to determine the best match between a target and search region is the sum-absolute-difference (SAD) algorithm [28]:

$$E_{m,n} = \sum_{i=1}^{l} \sum_{j=1}^{k} \left| X_{i,j} - Y_{i+m,j+n} \right| \tag{28}$$

where $E_{m,n}$ is the non-normalized error calculated over the target region and location m,n in the search region. The location m,n which has a minimum in $E_{m,n}$ corresponds to the new location of the target region. This algorithm requires only one arithmetic operation per pixel instead of eight with the correlation search algorithm. Experimental results also indicate that the SAD algorithm tracks speckle patterns as well as the correlation search algorithm [28].

2.3.2.4 Two-dimensional correlation search in the third moment domain

The correlation search algorithm and the SAD algorithm assume that any noise in the ultrasound imaging system is uncorrelated and will not affect the location where the correlation coefficient is maximum. If there is noise present, the pixel value in the images will consist of the desired signal plus the noise signal:

$$\begin{aligned} X(i,j) &= S(i,j) + W_1(i,j) \\ Y(i,j) &= S(i-m,\ j-n) + W_2(i,j) \end{aligned} \tag{29}$$

where X(i,j) are pixel values in the images X and Y, S(i,j) is the original signal location in image X, S(i-m, j-n) is the moved signal in image Y, and W_1 and W_2 are unknown noise sources. If W_1 and W_2 are correlated Gaussian noise, the correlation search algorithm will not work well because it will estimate the cross-correlation of both the signal and noise. Correlated Gaussian noise may exist in sections of the ultrasound imager, such as the receiver amplifier or A/D [29].

An important property of higher-order spectra is that for Gaussian processes all higher-order spectra of order greater than two are identically zero [29]. Kim, Kinter, and Greenleaf [29] have transformed the normalized correlation coefficient into the third moment domain which should suppress the effects of correlated Gaussian noise. The correlation coefficient can be transformed into the third moment domain by the following transformations:

$$XT(k,l) = E[X(i, j)\ X^2(i+k, j+l)] \qquad (30)$$
$$YT(k-u, l-v) = E[Y(i+u, j+v)\ X^2(i+k, j+l)]$$

where E[] is the expectation operator. The third moment correlation coefficient becomes

$$\rho_{u,v} = \frac{\sum_{k=1}^{N1}\sum_{l=1}^{N2}(XT(k,l)-\overline{XT})(YT(k-u,l-v)-\overline{YT})}{\sqrt{\sum_{k=1}^{N1}\sum_{l=1}^{N2}(XT(k,l)-\overline{XT})^2\ \sum_{k=1}^{N1}\sum_{l=1}^{N2}(YT(k-u,l-v)-\overline{YT})^2}} \qquad (31)$$

where XT and YT represent the transformed versions of X and Y and N1 and N2 are the dimensions of the target area. The location of the displaced target region is found at the u,v where $\rho_{u,v}$ is maximum.

The transformation into the third moment domain increases the computation of an already computationally extensive algorithm. A method of reducing the number of points that the target region is correlated in the search region can be accomplished by applying the constant brightness constraint of the optical flow algorithm [29]:

$$E_x u + E_y v + E_t = 0 \qquad (32)$$

The correlation coefficient $\rho_{u,v}$ need only be calculated at the locations u,v which satisfy Equation (32). This is illustrated graphically for a 5x5 target region and an 11x11 search region in Figure 23. The conventional correlation search algorithm requires 49 locations where the correlation coefficient is calculated; using Equation (32) as a constraint reduced the number of locations to 11.

2.4 Conclusion

Biological tissues can reflect ultrasound in many ways, depending on such factors as the healthiness and type of tissue, whether it is in motion, and the type of motion. By understanding how ultrasound interacts with tissues under various conditions, useful information about the tissue can be extracted from ultrasound echoes. This information can then be used to diagnose various medical disorders. The tissue motion assessment techniques reviewed in this chapter represent some of the latest methods under development which have the potential to be clinically useful.

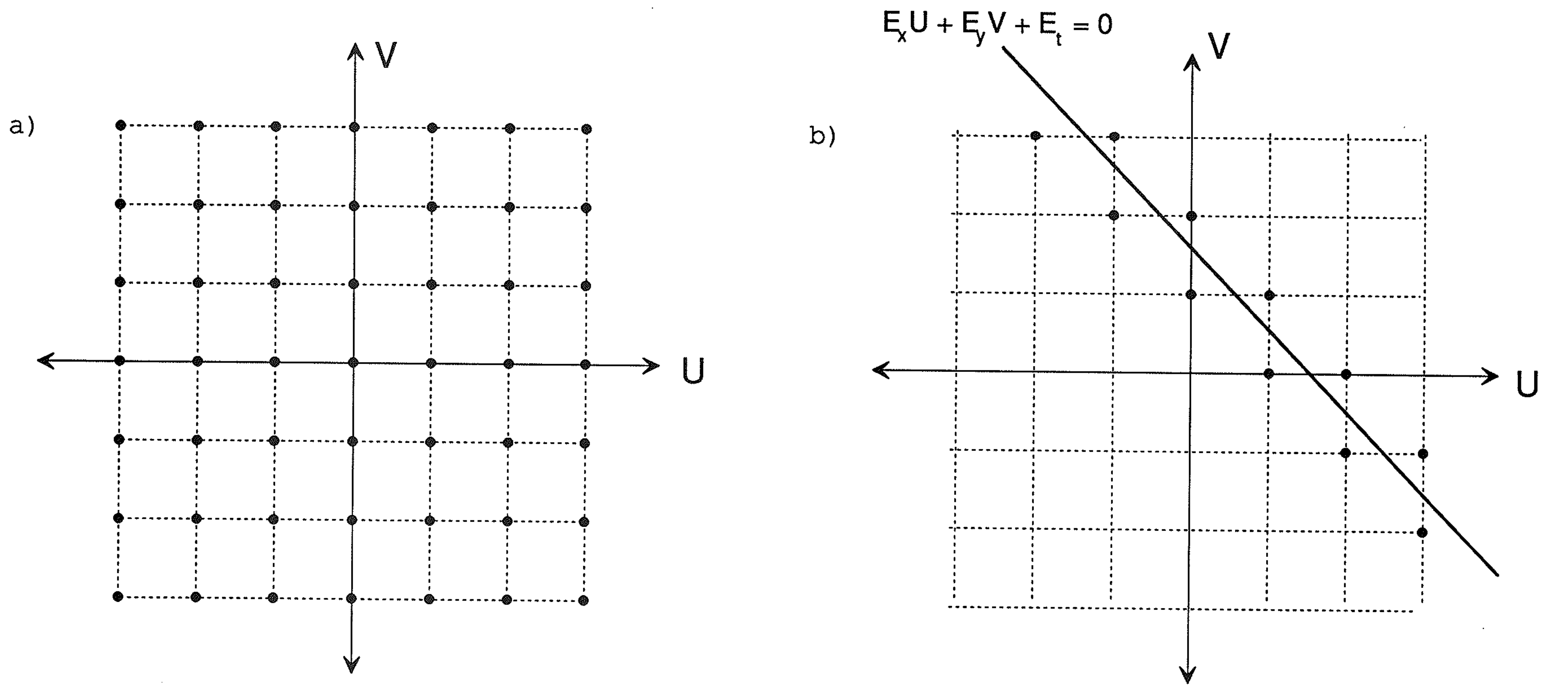


Figure 23 - Set of points where the correlation coefficient must be calculated for a) conventional and b) modified correlation search algorithms. (From Kim, Kinter, and Greenleaf, [29])

CHAPTER 3

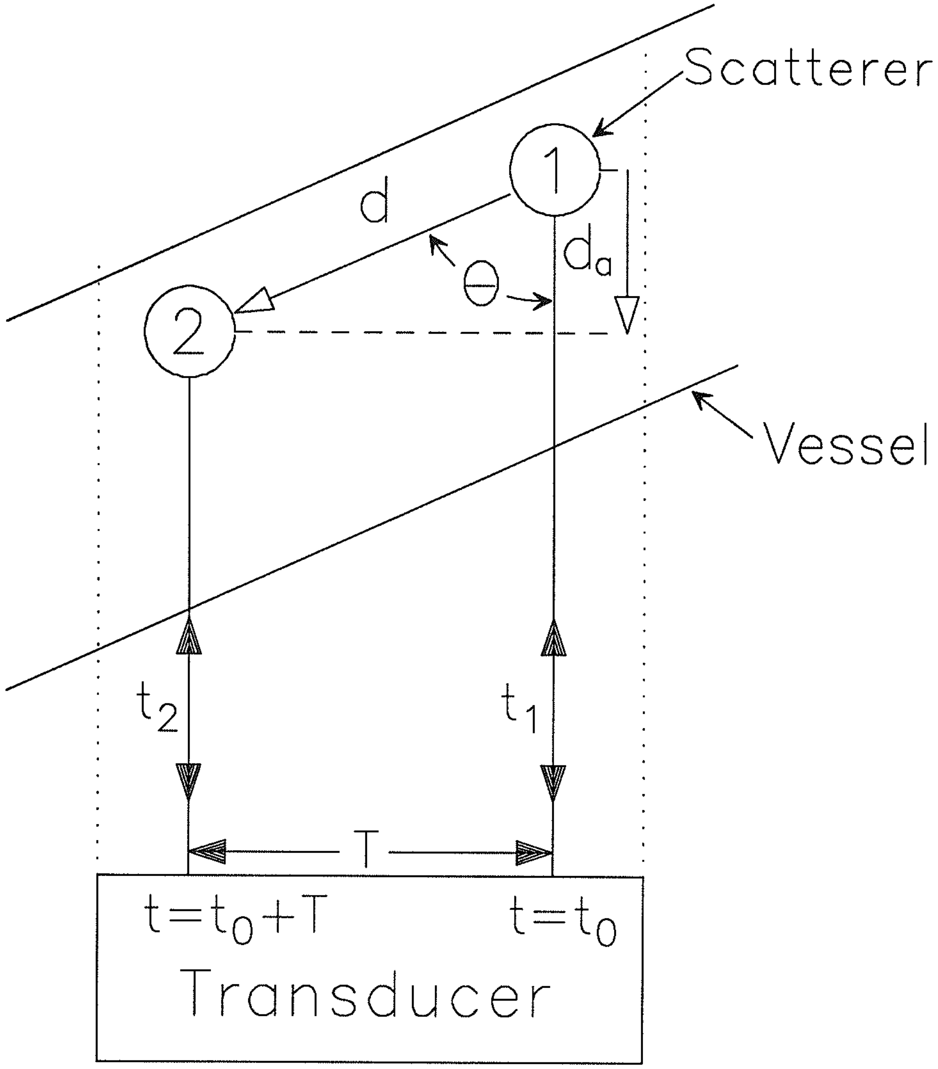
ULTRASOUND TIME-DOMAIN CORRELATION BLOOD FLOW MEASUREMENT TECHNIQUE

This chapter is intended to provide the background of the ultrasound time-domain correlation blood flow measurement technique. Brief comparisons are made to the Doppler technique and potential advantages over the Doppler technique are also pointed out where applicable.

3.1 Time-Domain Velocity Measurement Concept

The ultrasound time-domain flowmeter concept is illustrated in Figure 24. Here an ultrasound transducer is oriented at some angle θ with respect to a blood vessel. At time $t = t_o$, a blood cell scatterer is located in position 1. If an ultrasonic pulse is transmitted at time $t = t_o$, it will take a round trip time t_1 to leave the transducer, get reflected, and return to the transducer. If the next ultrasonic pulse is initiated T seconds later at time $t=t_o+T$, the scatterer will have moved to position 2 and the round trip transit time will be t_2. The distance d_a the scatterer has moved axially (in the direction of the ultrasound beam) can be calculated from the difference in transit times of the two ultrasound pulses:

$$d_a = (t_1 - t_2)c/2 \qquad (33)$$



$$d_a = \frac{(t_1 - t_2)C}{2}$$

$$d = \frac{(t_1 - t_2)C}{2\cos(\Theta)}$$

$$v_s = \frac{(t_1 - t_2)C}{2T\cos(\Theta)}$$

Figure 24 - Ultrasound time-domain flowmeter concept. The velocity of a scatterer can be calculated from the transit times of two ultrasonic pulses reflected from the scatterer at two positions along the vessel.

where c is the speed of sound in the medium. The distance d the scatterer has moved down the vessel is $d_a\cos(\theta)$, where θ is the transducer measurement angle (also called the Doppler angle). Since velocity is distance/time, the scatterer velocity is (assuming $V_s \cos(\theta) << c$)

$$V_s = \frac{(t_1 - t_2)c}{2\ T\ \cos(\theta)} \tag{34}$$

This time-domain equation is actually very similar to the Doppler equation (see Equation (1)) except that it has a change in time in the numerator instead of a change in frequency, and the pulse repetition period in the denominator instead of the transmitted frequency. The change in time $t_1 - t_2$ is referred to as the time shift in this thesis and is denoted by the variable τ. Calculation of the scatterer velocity in the time domain instead of the frequency domain has theoretical advantages which will be discussed later.

3.2 Reflection of Ultrasound by Blood Cell Scatterers

In the real-life situation, human beings do not have only one blood cell moving through their arteries or veins and the actual ultrasonic echo is due to all of the scatterers within the ultrasound beam, as illustrated in Figure 25. In this figure E_1 is the echo due to volume 1 (V_1), which has almost moved out of the ultrasound beam due to motion of the scatterers. E_2 is the echo due to volume 2 (V_2), which is shown totally within the

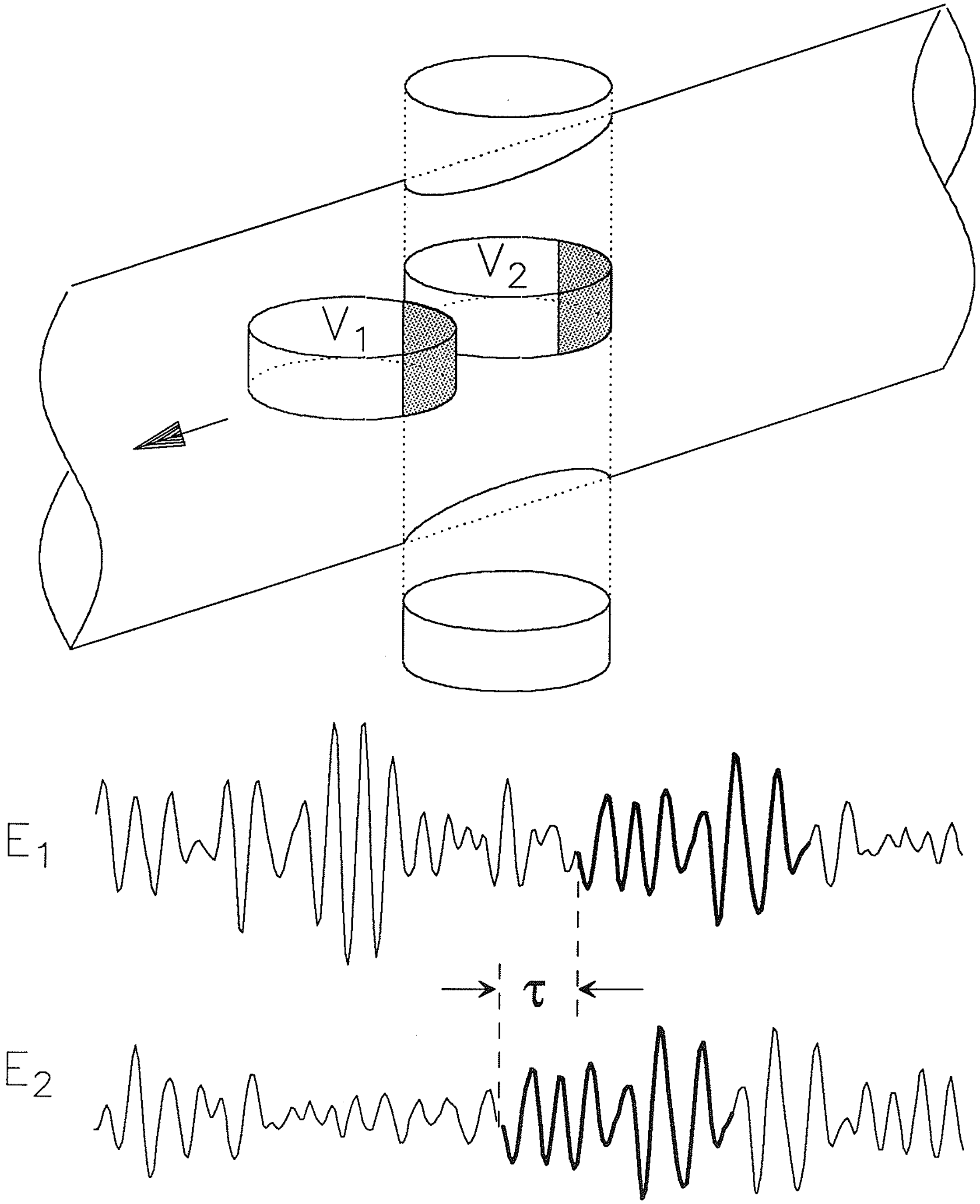


Figure 25 - The actual ultrasonic echoes are due to all scatterers within the ultrasonic beam. E_1 is the echo due to volume 1, which has almost moved out of the beam. E_2 is the echo due to volume 2, still within the beam. The common sections of V_1 and V_2 will produce similar sections of echo in E_1 and E_2.

ultrasound beam. If the pulse repetition period is chosen such that some of the original scatterers remain common to both pulses (shaded areas of V_1 and V_2), then these common sections of volume will produce similar sections of echo in E_1 and E_2. These similar sections of echo will be displaced in time from each other by the time shift τ. Since blood reflects ultrasound in a random nature [7], any small volume of blood cell scatterers will have its own unique ultrasonic footprint, and the common sections of echo will represent reflections from the same group of blood cell scatterers. Equation (34) can then be used to calculate the velocity of a particular blood scatterer group if τ can be determined.

3.3 Calculation of the Time Shift via Correlation

To calculate the time shift between two similar sections of echoes, the echoes are correlated with each other. This process is illustrated in Figure 26 for a point scatterer. If $E_1(t)$ and $E_2(t)$ represent signals received at different times from a moving scatterer, the correlation can be pictured as shifting E_1 back in time by some value of τ and multiplying by E_2 to produce the correlation coefficient $R(\tau)$. Mathematically this can be expressed as

$$R(\tau) = \Sigma_t E_1[t+\tau]E_2[t] \qquad (35)$$

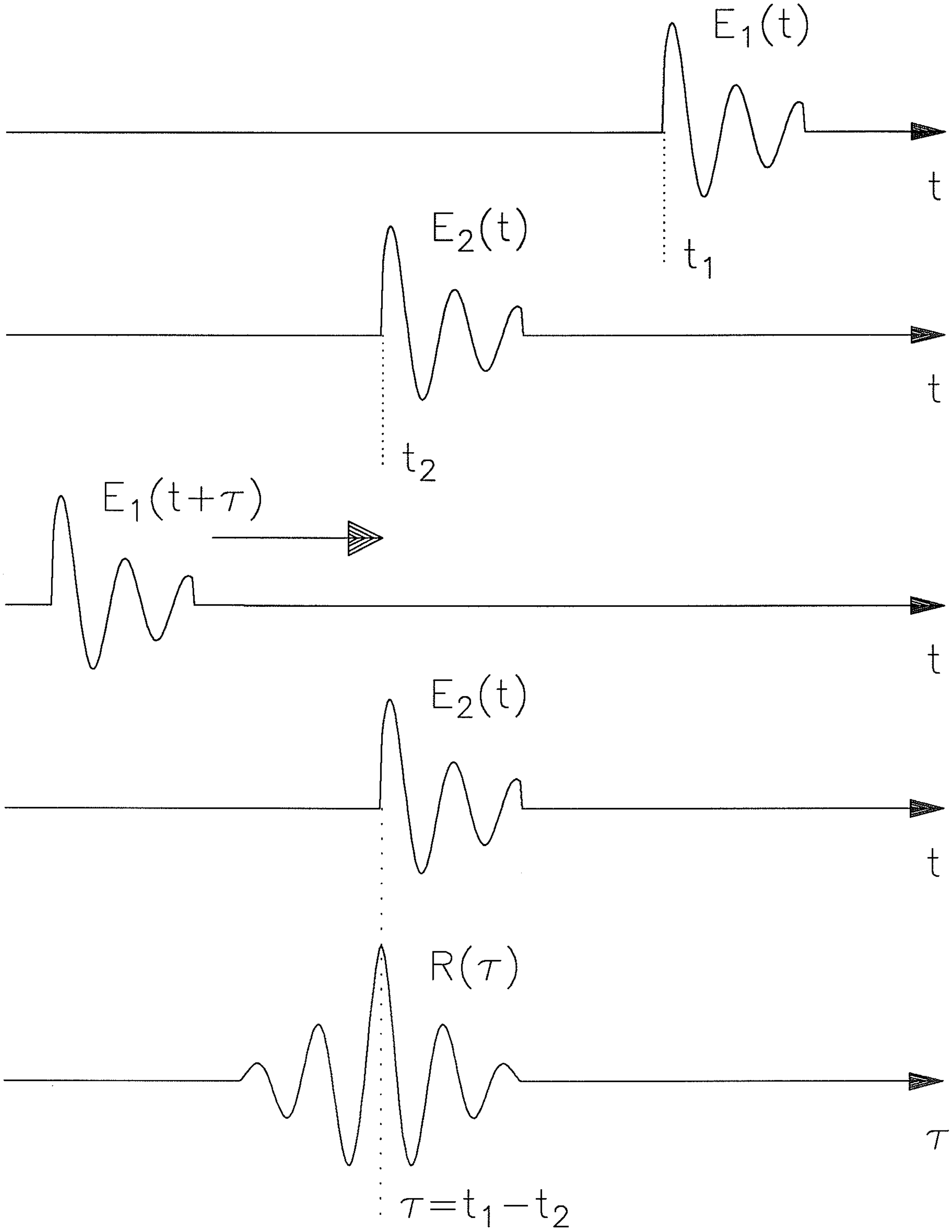
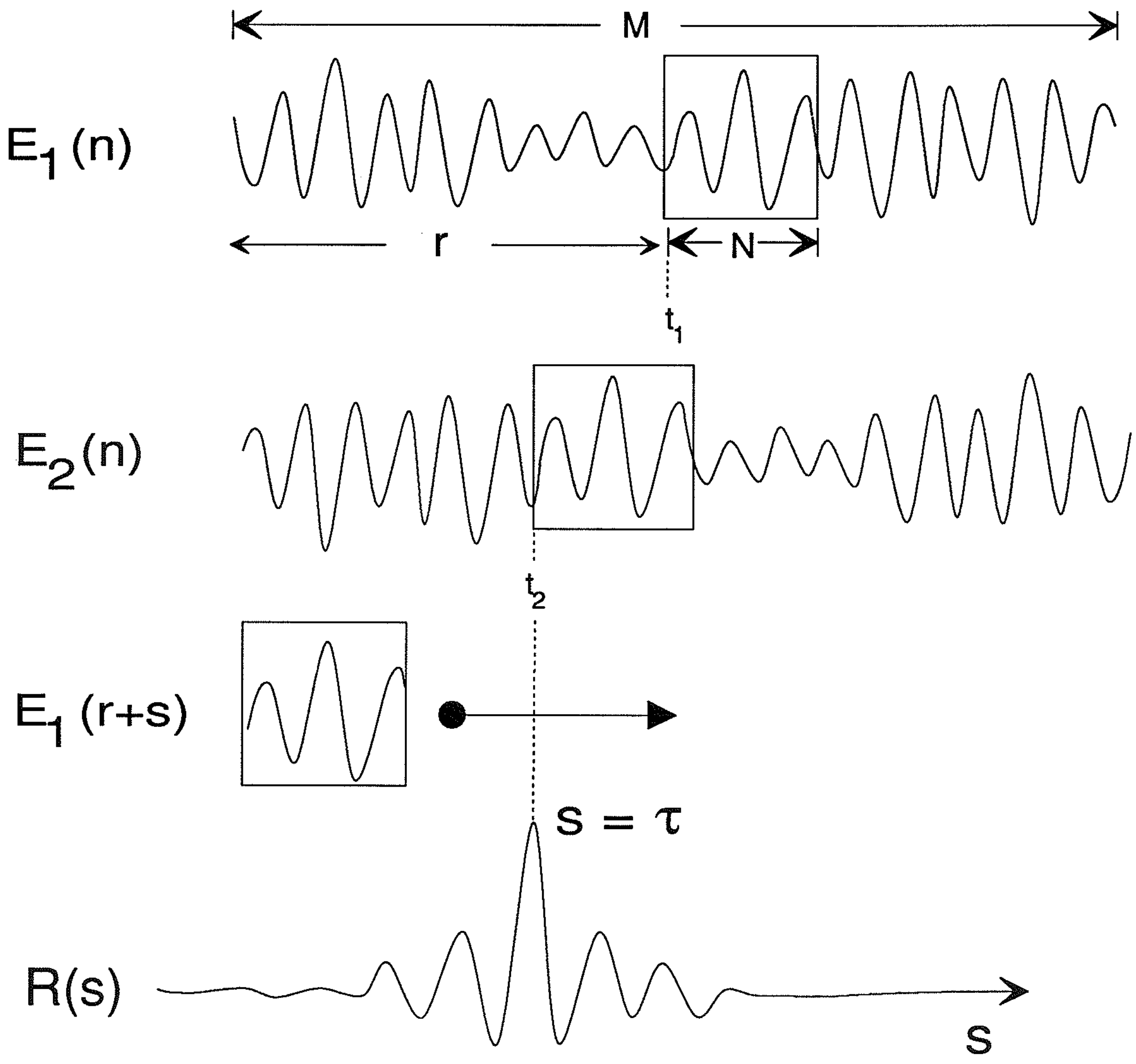


Figure 26 - The correlation of two echoes consists of shifting one echo back in time by τ and multiplying it by the other to produce the correlation coefficient $R(\tau)$. The value of τ producing a maximum $R(\tau)$ corresponds to $\tau = t_1 - t_2$.

The value of τ which produces a maximum in the correlation function $R(\tau)$ corresponds to the time shift $t_1 - t_2$ and Equation (34) can be used to calculate the scatterer velocity.

A major advantage of the time-domain correlation procedure over Doppler is that the correlation function is not affected by the frequency-dependent attenuation of the intervening tissues. This is because each echo passes through the same intervening tissue and is affected in exactly the same manner as all other echoes, which does not affect the correlation function. The correlation method will also have an inherently better signal to noise ratio since all of the energy in each echo is used. Doppler systems mix the incoming RF signal to determine the Doppler frequency and some of the energy is lost to sidebands due to the mixing process [7].

As stated earlier, the reflected ultrasound echoes will not be due to a point source but due to all of the scatterers in the ultrasound beam. The correlation process for many scatterers within the beam is shown in Figure 27. The RF echoes E_1 and E_2 are digitized for a digital length of M samples. A section of digital length N samples is windowed out from E_1 at a digital length r from the beginning of the echo. This window is correlated with equivalent length windows at different locations r+s ($-r \leq s \leq M-N$) along E_2 where s is the current correlation window location in E_2 with respect to r of E_1. The value of s where the normalized correlation function R(s) is maximum corresponds to the



$$R(s) = \frac{\sum_{i=0}^{N-1} E1(r+i)\, E2(r+i+s)}{\sqrt{\sum_{j=0}^{N-1} [\, E1(r+j)\,]^{2} \sum_{k=0}^{N-1} [\, E2(r+k+s)\,]^{2}}}$$

Figure 27 - The correlation of 2 digitized echoes consists of windowing out a desired section of one echo and correlating it at different locations s along the other echo. The value of s producing maximum R(s) corresponds to s = τ.

time shift in units of the sampling period. The normalized correlation coefficient R(s) is given by the equation

$$R(s) = \frac{\sum_{i=0}^{N-1} E_1(r+i)E_2(r+i+s)}{\sqrt{\sum_{j=0}^{N-1} [E_1(r+j)]^2 \sum_{k=0}^{N-1} [E_2(r+k)]^2}} \tag{36}$$

and produces a value between -1 and 1, where a value of 1 corresponds to exact overlapping of the echoes.

The digital lengths are related to actual physical distances by the equation

$$d_{phys} = pT_sc/2 \tag{37}$$

where p is the digital length in number of sample points, T_s is the sampling period of digitization, and c is the speed of sound in the medium.

3.4 One-Dimensional Velocity Measurements

If the digitized echo length M is chosen to correspond to at least the diameter of the vessel, the velocity at different points along the vessel can be sampled by varying the window location r of the first echo E_1. Equation (37) can be used to calculate the physical distance (referred to as the range) from the transducer that r corresponds to and a one-dimensional velocity vs. range profile across the vessel can be generated. Figure 28 illustrates a one-dimensional flow velocity measurement and velocity profile for a fully developed laminar flow within a vessel. A fully developed laminar flow has been used in a blood flow phantom system to validate the UTDC technique [10], and all of the blood flow phantom experiments described in this thesis have been made with fully developed laminar flow within a vessel. The front and rear vessel wall locations (d_f and d_r in Figure 28) can be determined at the ranges where the velocity profile becomes zero. One-dimensional velocity measurements can be made at different angles in the cross-sectional plane of the vessel, as shown in Figure 29. If the one-dimensional scan corresponds to the transducer aimed directly through the center of the vessel (angle α_{max} in Figure 29), the volumetric flow can be calculated from the vessel wall locations d_f and d_r and the maximum velocity V_{max} of the flow profile if the measurement angle θ is known. This is described in greater detail in Chapter 6, Section 2.

3.5 Two-Dimensional Velocity Measurements

As shown in Figure 29, the velocities at different points along the cross-sectional area of the vessel can be sampled by performing one-dimensional axial velocity scans at different scan

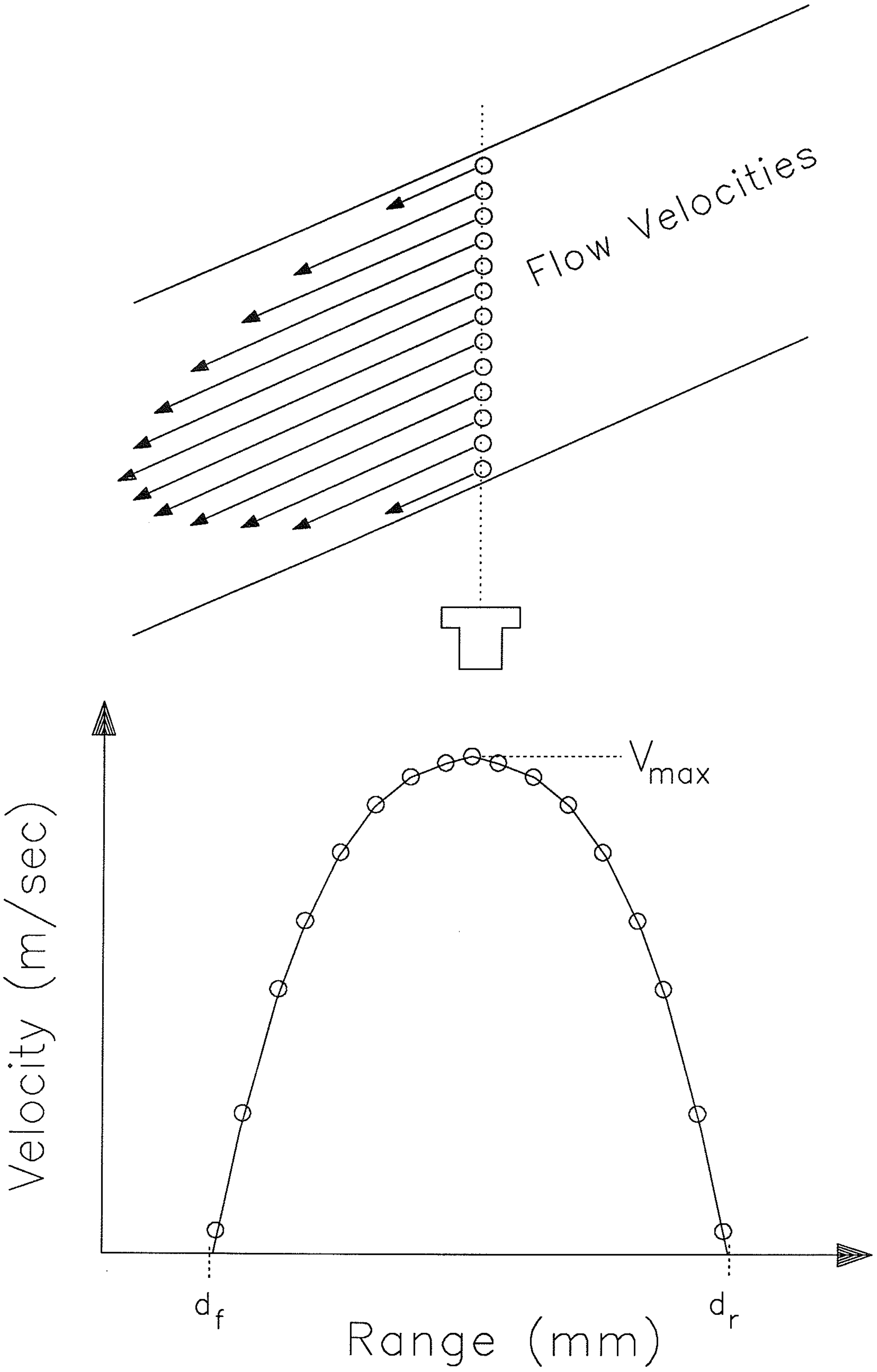


Figure 28 - One-dimensional axial flow velocity measurement.

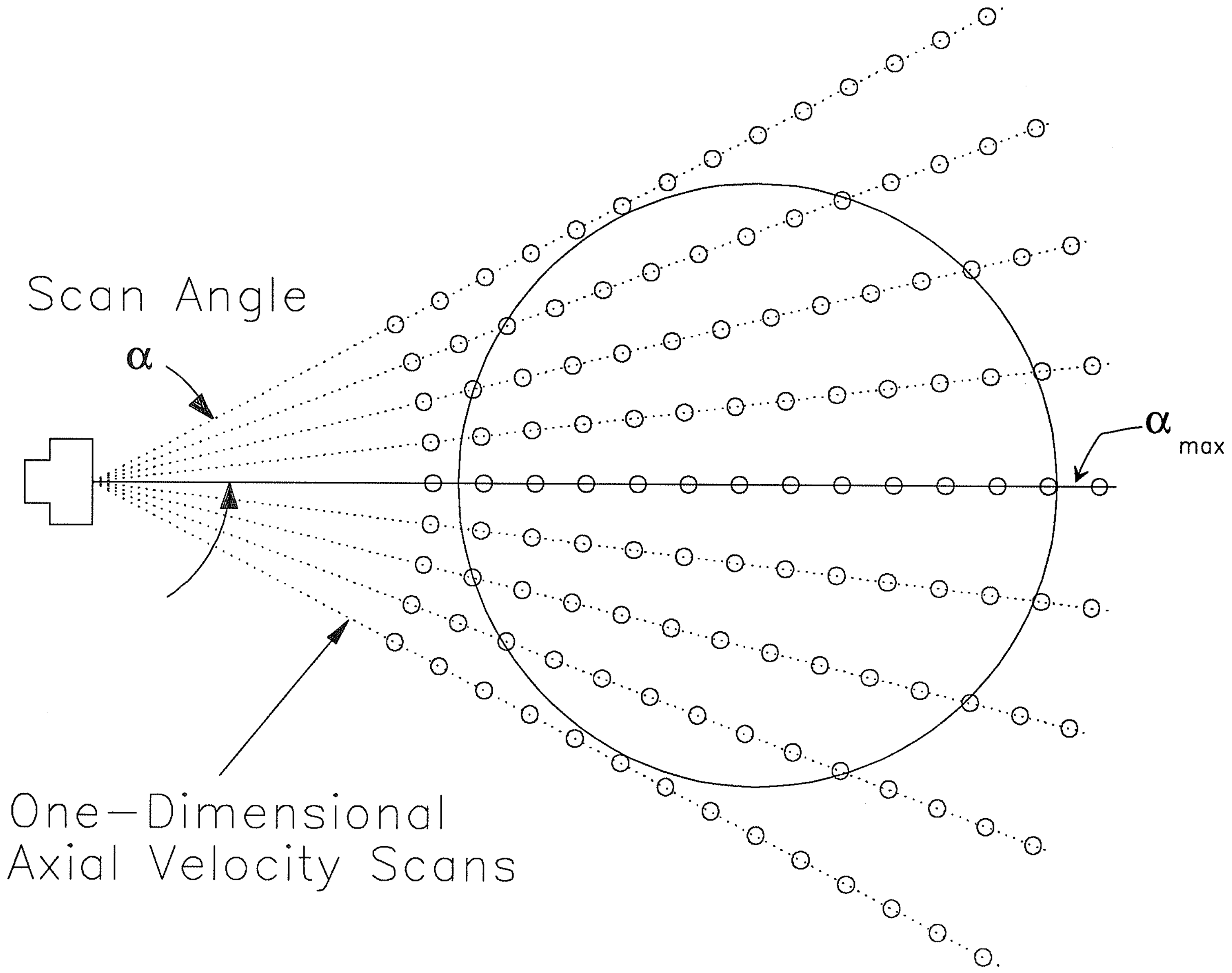


Figure 29 - Two-dimensional flow velocity measurement.

angle αs. If a contour plot is made where each contour represents a constant velocity level, a set of concentric ellipses will be formed as shown in Figure 30. The outermost ellipse represents the location of the vessel walls and the transducer measurement angle θ can be calculated from the ratio of the major and minor axes of the ellipses using the equation

$$\theta = \frac{1}{M} \sum_{n=0}^{M-1} \sin^{-1} \left[\frac{A_n}{B_n} \right] \tag{38}$$

where A_n and B_n are the minor and major axes of the, respectively, of the nth ellipse. The volumetric flow Q can be estimated by numerically integrating over the constant velocity ellipses [10]. The data from a two-dimensional measurement can also be plotted as a color flow image or presented in a three-dimensional format (Chapter 5).

3.6 Weighted Axial Velocity Measurement

It has been shown in [8] that the precision of the UTDC technique is dependent on the time shift τ, which is directly determined by the PRF. For a given velocity, a faster PRF will produce a smaller time shift, since the scatterers will have less time to move between pulses. Since the blood scatterer velocity may change, such as with pulsatile flow in the human body, a single PRF may have optimal precision at one scatterer velocity

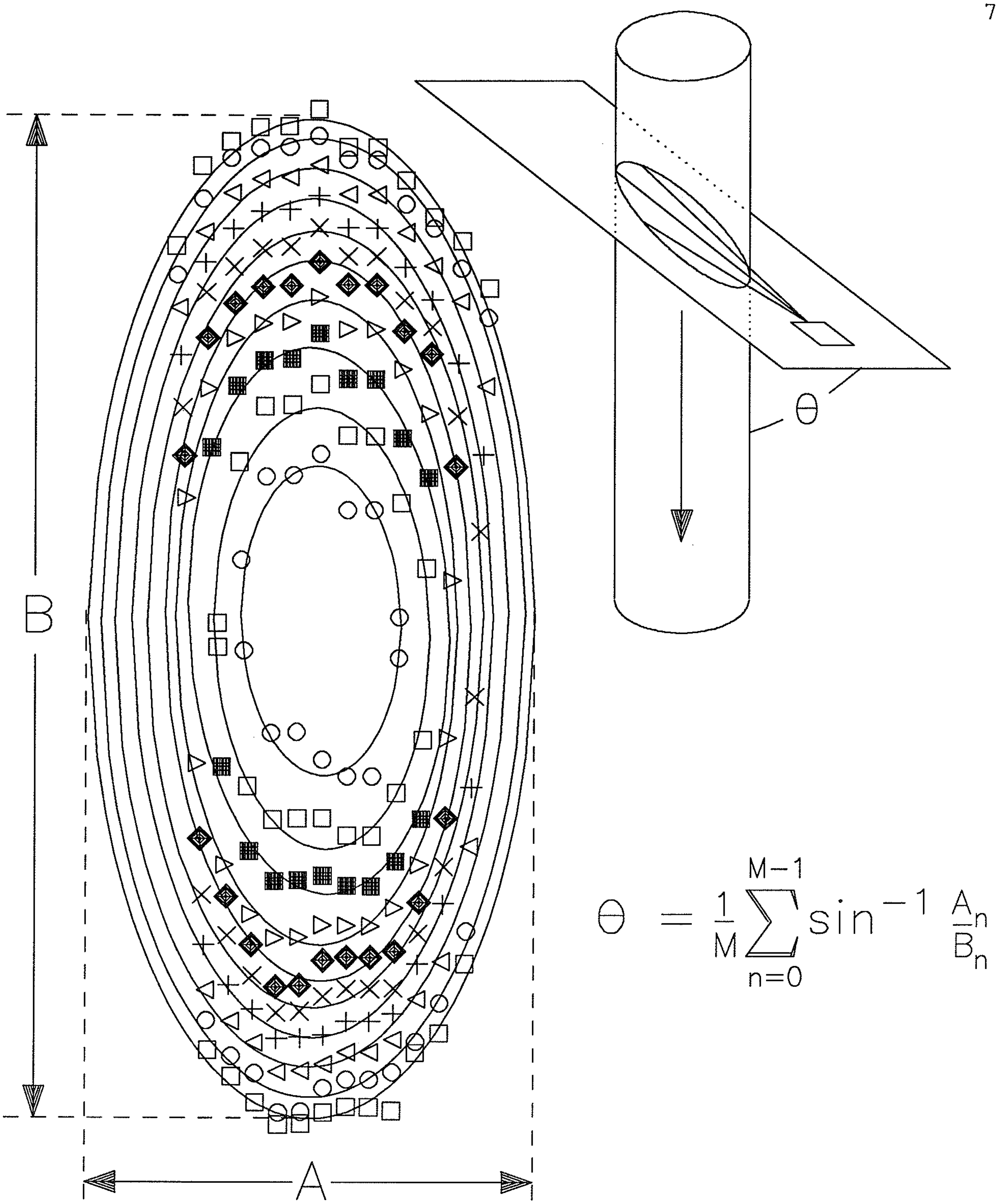

θ
B
$\theta = \frac{1}{M}\sum_{n=0}^{M-1}\sin^{-1}\frac{A_n}{B_n}$
A

Figure 30 - The transducer measurement angle can be calculated from the minor and major axes of ellipses' curve fit to the velocity data.

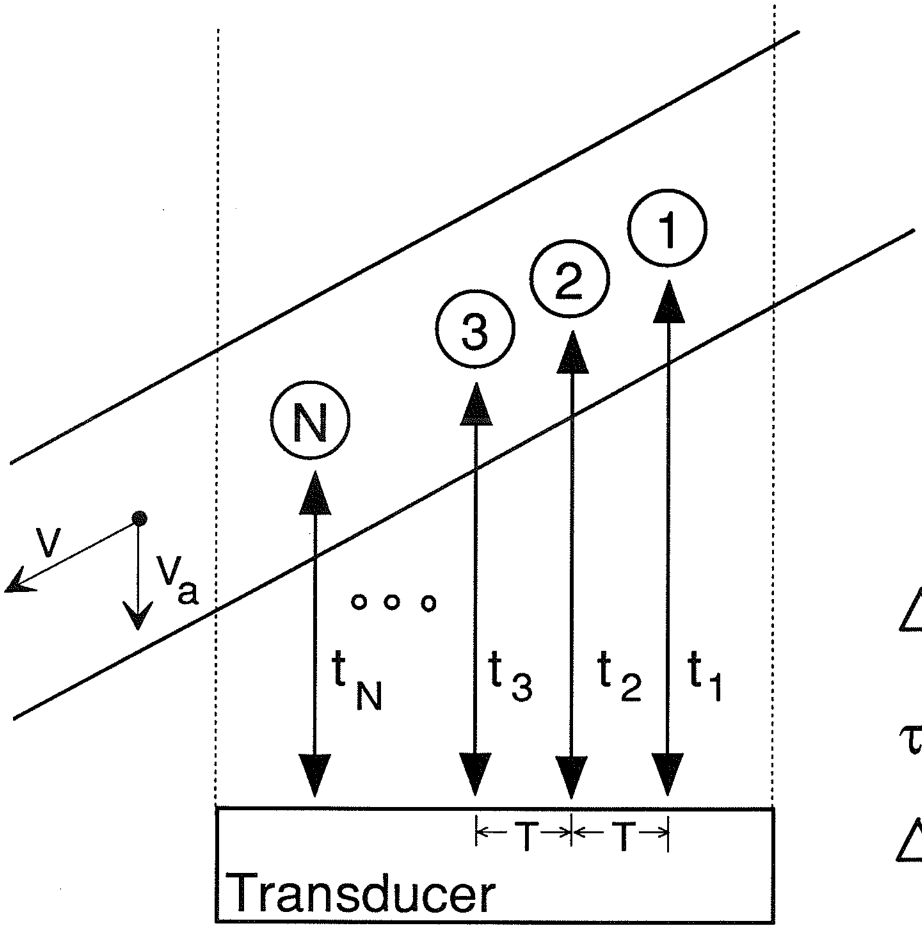
but poor precision at a faster or slower velocity. A method of effectively varying the PRF without physically doing so is to exploit the fact that the scatterers remain within the ultrasound beam for more than just two pulses, as shown in Figure 31. If the scatterer remains within the beam for N pulses, any two echo pairs can be used as an independent velocity measurement. The variable Δ has been defined as the echo pair spacing. A $\Delta = 1$ corresponds to correlations between adjacent echoes, such as between echoes 1 and 2, 2 and 3, etc.; a $\Delta = 2$ corresponds to every other echo, such as between echoes 1 and 3, 2 and 4, etc., up to a maximum echo spacing $\Delta_{max} = N-1$. Thus for any two echoes i and j (j>i), $\Delta = j-i$ and the time shift $\tau_\Delta = t_i - t_j$. The axial velocity $V_{a\Delta}$ can be calculated for a given echo spacing by

$$V_{a\Delta} = \frac{c\ \tau_\Delta}{2\ \Delta\ T} \tag{39}$$

where c is the speed of sound and T is the physical pulse repetition period. A weighted velocity average over N echoes can be calculated by the equation

$$V_a = \sum_{\Delta=1}^{\Delta max} W_\Delta V_{a\Delta} \tag{40}$$

where V_a is the final axial velocity estimate and W_Δ is a weighting factor determined by the variance of the individual $V_{a\Delta}$ estimates. Those estimates with a low variance are weighted more heavily than



$$\Delta = j\text{-}i \quad j > i$$

$$\tau_\Delta = t_i - t_j$$

$$\Delta_{max} = N - 1$$

$$V_{a\Delta} = \frac{c\tau_\Delta}{2\Delta T}$$

$$V_a = \sum_{\Delta=1}^{\Delta_{max}} W_\Delta V_{a\Delta}$$

Figure 31 - Axial velocity determination using a weighted average over the different echo spacings.

those with a higher variance. In this manner the PRF with the optimal precision can be statistically determined and weighted accordingly to produce an optimal axial velocity estimate [9].

CHAPTER 4

BLOOD FLOW PHANTOM

This chapter describes the blood flow phantom system used to validate the ultrasound time-domain correlation technique under pulsatile flow conditions as well as validation of the temporary and real-time systems described in Chapters 7 and 8.

4.1 Flow Generation

Figure 32 illustrates the blood flow phantom system. It is a modification of the one used by Embree [9] which was only capable of producing constant flow. The modified blood flow phantom system is capable of producing both constant and pulsatile flow through a section of dialysis tubing.

4.1.1 Constant flow generation

The constant flow rate is set by adjusting the height H between the upper and lower reservoirs. Gravity will cause fluid to flow from the upper reservoir through the section of dialysis tubing to the lower reservoir, and from there to the return reservoir. A peristaltic pump moves the fluid back to the upper reservoir. The fluid is pumped from the bottom of the return

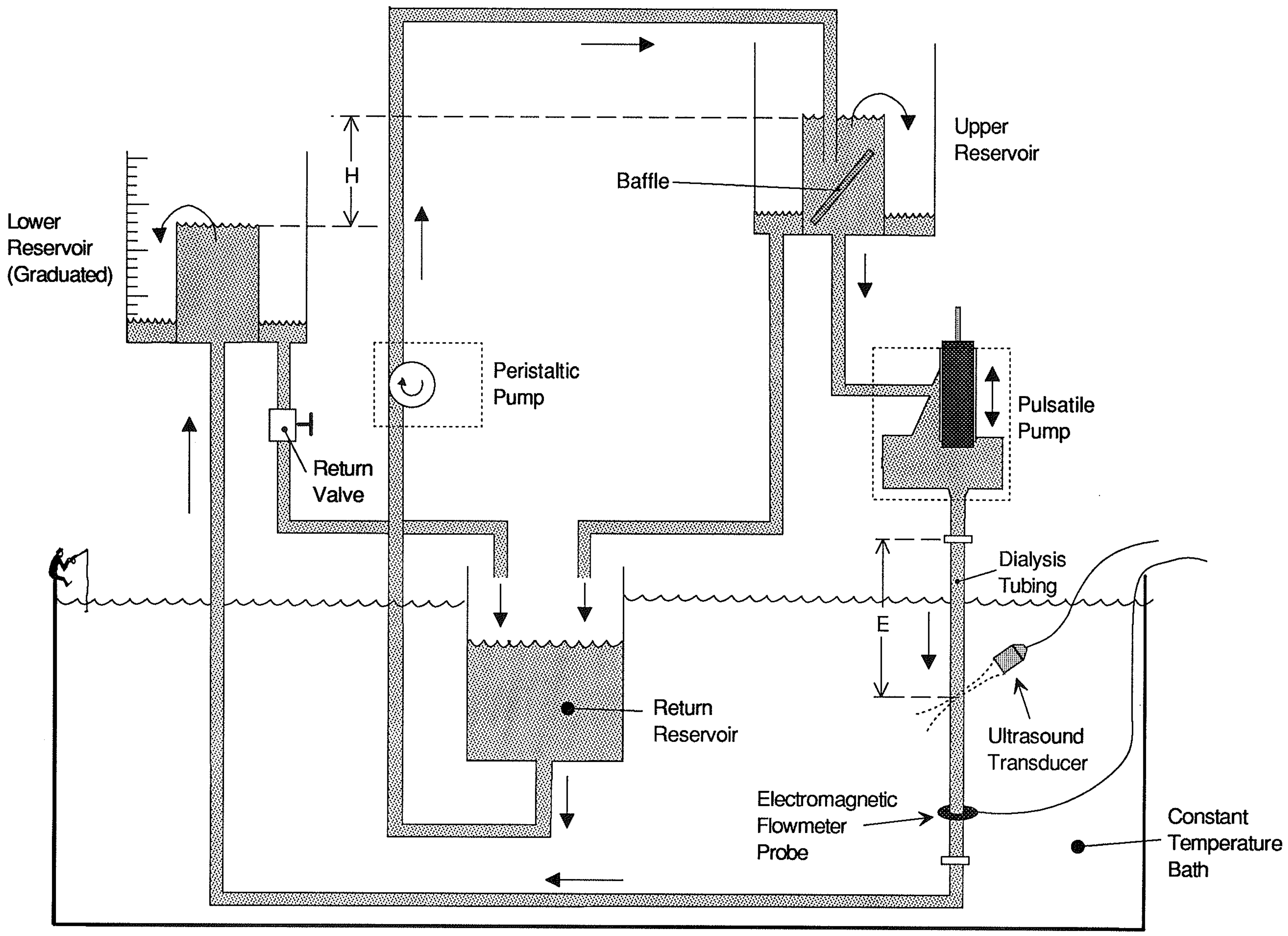


Figure 32 - Blood flow phantom system. Continuous and pulsatile flow can be generated.

reservoir to help keep the Sephadex in the Sephadex-water fluid solution (described in Section 4.4) from settling out. The constant flow system is described in detail in [9].

4.1.2 Pulsatile flow generation

The blood flow phantom of [9] has been modified by placing a pulsatile pump in series between the upper reservoir and the section of dialysis tubing. When the pulsatile pump is off, only constant, non-time varying flow is present. When the pulsatile pump is on, a peak-to-peak flow component oscillates around an average flow component. Figure 33 illustrates the pulsatile pump system. It consists of a pulsing unit, which contains a fluid chamber and a syringe that can move in and out of the chamber (details of the pulsing unit can be found in Appendix A). The continuous flow component enters the pulsing unit near the top, fills the fluid chamber, and then passes out the bottom to the dialysis tubing. The motion of the syringe is controlled by a motor and gearbox assembly. The gearbox translates the circular motion of the motor to the translational back-and-forth motion of the syringe. When the syringe is pulled out of the chamber, the continuous flow component is reduced by the volume pulled back by the syringe; and when it is pushed into the chamber, the continuous flow component is increased by the volume displaced in the chamber by the syringe. The volume displacement of the syringe is set at ±1.4 ml. The frequency of pulsation can be controlled by the

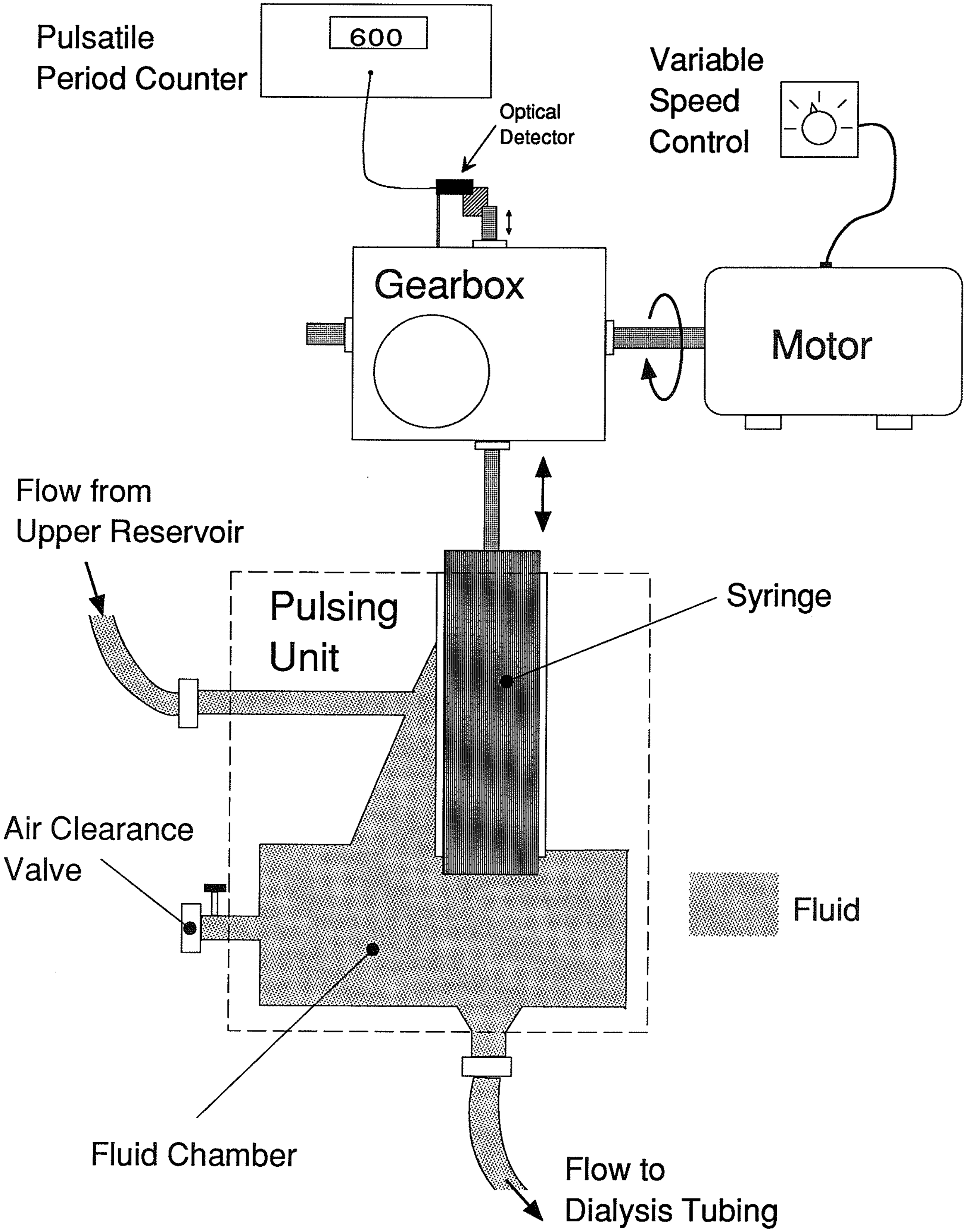


Figure 33 - Pulsatile pump system. A variable speed motor moves a syringe in and out of a chamber producing pulsatile flow.

motor, which has a variable speed control. The volume displacement of the syringe is then

$$v(t) = 1.4 \sin(\omega t) \text{ ml} \tag{41}$$

where v is the volume displaced in the fluid chamber and ω is the translational frequency of the syringe. The period of the pulsing of the syringe is measured by an inexpensive custom-built pulsatile period counter (details described in Appendix B). It consists of an infrared optical detector and an interrupter connected to the translational shaft of the gearbox. The interrupter interrupts the infrared beam in the optical detector whenever the syringe is pulled all the way back, which happens once every pulse. The pulsatile period counter then displays the time between pulses.

4.2 Flow Velocity Distribution in the Dialysis Tubing

All ultrasonic flow measurements are made when the fluid flow is passing through the section of dialysis tubing. A minimum entrance length E exists between the entrance of flow into the dialysis tubing and the ultrasonic measurement point to insure that fully developed laminar flow exists for all flow rates, both continuous and pulsatile. For most experiments the dialysis tubing used was 6.5 mm SPECTRA/POR D1615-4.

4.3 Independent Measure of Volumetric Flow

The volumetric flow passing through the dialysis tubing can be measured independently by two means. For constant flow experiments, the volumetric flow can be estimated by closing the return valve on the lower reservoir, which is graduated, and measuring the time it takes for a given volume in the reservoir to be filled. This has been referred to as the hydrodynamic flow rate and has an accuracy of ±0.7% [9]. For pulsatile experiments this method cannot be used and a Zepeda Instruments SWF-5RD electromagnetic flowmeter with an accuracy of ±3.0% [30] is used as the standard to which the ultrasound measurement is compared.

4.4 Blood Mimicking Fluid

The blood-mimicking fluid used in the phantom is either distilled water or normal saline mixed with Sephadex (G-50; 20-80 μm diameter, Pharmacia Fine Chemicals, Uppsala, Sweden). This mixture has been determined to reflect ultrasound in a similar manner as flowing blood [9]. For constant flow experiments the distilled water-Sephadex mixture is used; for pulsatile experiments, the normal saline-Sephadex mixture must be used because the electromagnetic flowmeter requires the fluid to be conductive. The fluid in the constant temperature bath must match the fluid in the flow phantom (either distilled water or normal saline).

4.5 Other Modifications

The blood flow phantom has also been modified to allow laminar flow to be produced in two vessels (sections of dialysis tubing) placed next to each other, for different size vessels, and for vessels with a section occluded. These modifications involve only the segment of the phantom where the dialysis tubing is placed and are explained in greater detail in Chapter 5, Sections 2.1 and 2.3, and Chapter 8, Section 2.6.

CHAPTER 5

PREVIOUS SYSTEM MODIFICATIONS AND CONSTANT FLOW MEASUREMENTS

This chapter describes measurements and modifications made to the previously existing time-domain correlation system built by Paul Embree [9]. This modified system was used to perform constant flow experiments not performed in [9] as well as pulsatile flow experiments described in Chapter 6.

5.1 Original Time-Domian Correlation System

The original time-domain correlation system described in [9] consisted of an ultrasound data acquisition system (UDAS) interfaced to a VAX 11/730 computer. This system was awkward to use and had minimal graphical capability. The VAX 11/730 was a multi-user lab computer and the time spent in processing of the time shifts provided by UDAS varied greatly depending on the number of users on the system. Graphics capability was limited to a dot matrix line printer connected to the VAX. The UDAS consists of a 50 MHz A/D, 384k of memory, and a hardware correlator based on a TRW multiplier board. It takes UDAS approximately 90 sec to calculate and transfer the time shifts for 384 echoes to the VAX. There were no modifications made to UDAS for constant flow experiments. The modifications made to UDAS for pulsatile flow experiments are described in Chapter 6.

5.1.1 Replacement of the VAX

The main modification of the previous system was the replacement of the VAX and modifications to the software. The VAX 11/730 computer was replaced by a dedicated COMPAQ 386 personal computer running at 20 MHz. The COMPAQ was equipped with both 387 and Weitek math coprocessors and a VGA graphics display. The original flow measurement software that was written in FORTRAN on the VAX was modified and translated into the C language and compiled using Metaware's High-C 386 compiler. This provided the graphical capability of producing immediate (as compared to waiting for the line printer) high-resolution color plots on the computer screen. It also improved processing speed significantly over the VAX. The elliptical curve-fitting routine used to calculate the volumetric flow as described in [10] took about two and a half min to process on the VAX; using the same dataset took ten sec on the COMPAQ using the Weitek math coprocessor (the same calculations took 15 sec using the 387 coprocessor).

5.1.2 Software modifications

The basic velocity calculation algorithms were all preserved from the original FORTRAN programs. Software was added which provided the capability of producing the same plots originally plotted on the line printer on the computer screen. Additional software was written which is capable of displaying two-dimensional velocity data in both a color flow image and in a three-dimensional perspective.

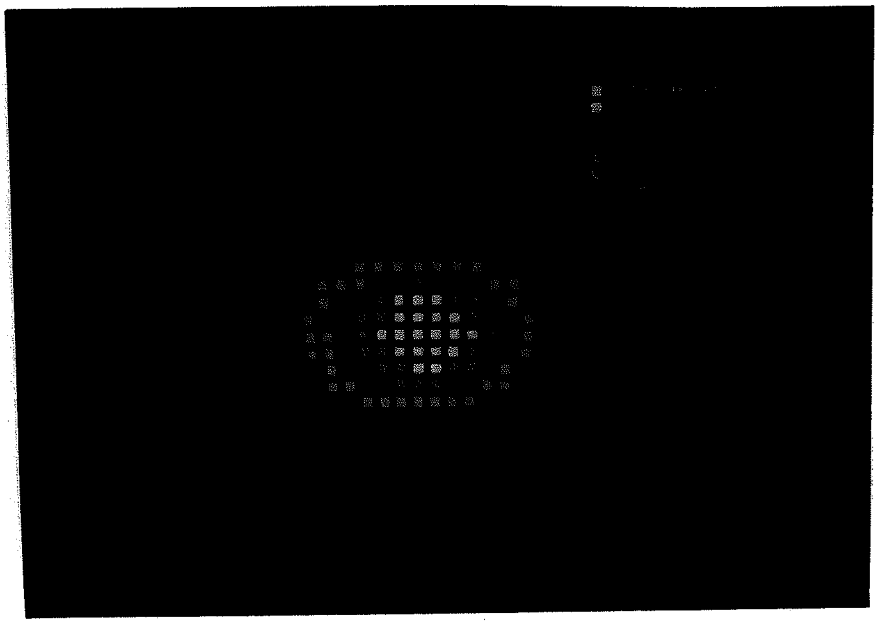
5.1.2.1 Color flow images

Figure 34a) shows a color flow image of fully developed laminar flow moving through the 6.5 mm dialysis tubing. The raw, angle uncorrected data are shown, resulting in the elliptical shape. Each color represents a different velocity range and the x and y axes of the plot represent distance from the transducer. The color white indicates the maximum velocity and the number displayed at the blue legend (0.50 cm/sec) indicates the minimum displayed velocity. All velocities below this cutoff are not displayed. The colors blue through yellow on the color velocity scale are assigned equal ranges where the range increment is equal to the maximum velocity minus the cutoff divided by 7 (since there are 7 available colors).

5.1.2.2 Three-dimensional perspective of flow

Figure 34b) shows a three-dimensional perspective of the same data in figure 34a). The vertical axis represents velocity and the horizontal axis represents distance from the transducer. This type of image is useful in visualizing different flow patterns due to obstructions or non-symmetric vessel shapes.

a)



b)

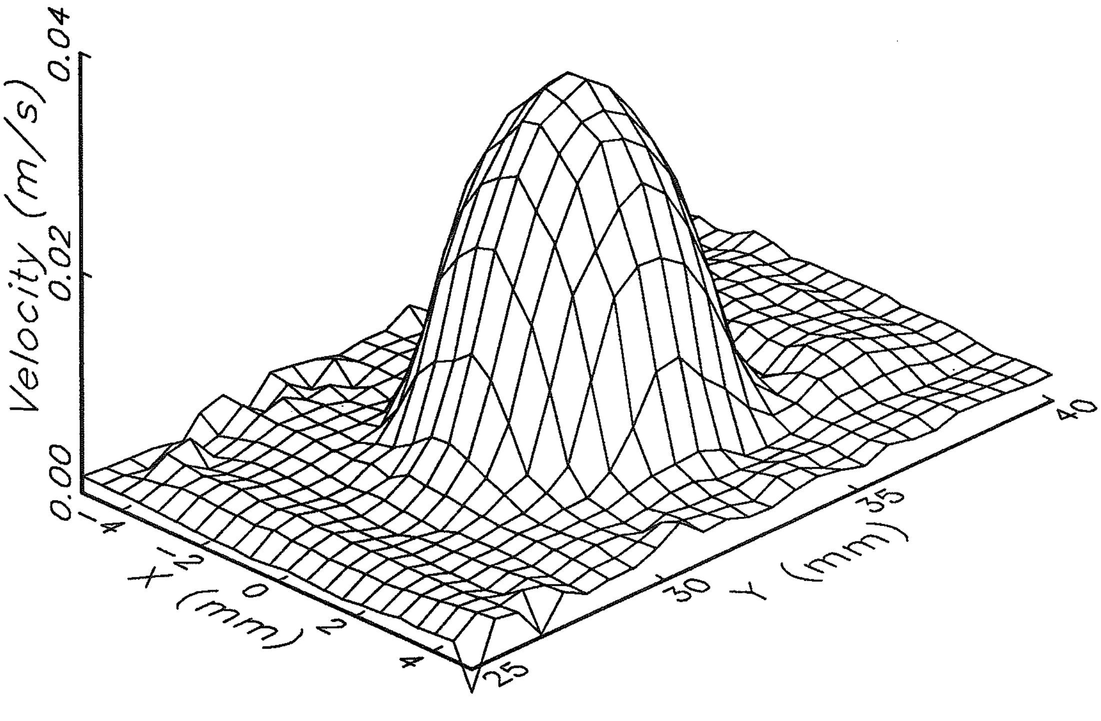


Figure 34 - a) Color flow profile of flow velocities in the 6.5 mm dialysis tubing. The cutoff (minimum plotted velocity) level is indicated by the dark blue color. b) Three-dimensional perspective of flow within the vessel.

5.2 Constant Flow Experiments

Many constant flow experiments identical to the ones performed in [10] were performed to validate the UDAS-COMPAQ system. In addition, experiments not performed earlier have also been made. These consisted of measurements with non-symmetric flow through the vessel and with two vessels placed next to each other.

5.2.1 Non-symmetric flow measurement

In the real-life situation the blood vessel shape may not be round or there may be occlusion or obstructions in the vessel. To see if the UTDC technique could image and estimate the flow through a partially occluded vessel, an obstruction was placed on one side of the vessel as shown in Figure 35. The obstruction used was a 3 mm diameter wire which was placed at the far side of the vessel (with respect to the transducer) and spanned the length of the dialysis tubing. This type of obstruction should still produce laminar flow but will change the distribution of flow within the vessel (it will no longer be axially symmetric and may not be parabolic). Figure 36 shows the resulting color flow image and three-dimensional perspective of velocities within the vessel. A total of 22 one-dimensional A-scans were made, with 17 of them intersecting the vessel. The color flow image in Figure 36a) shows quite clearly the location of the obstruction and how the

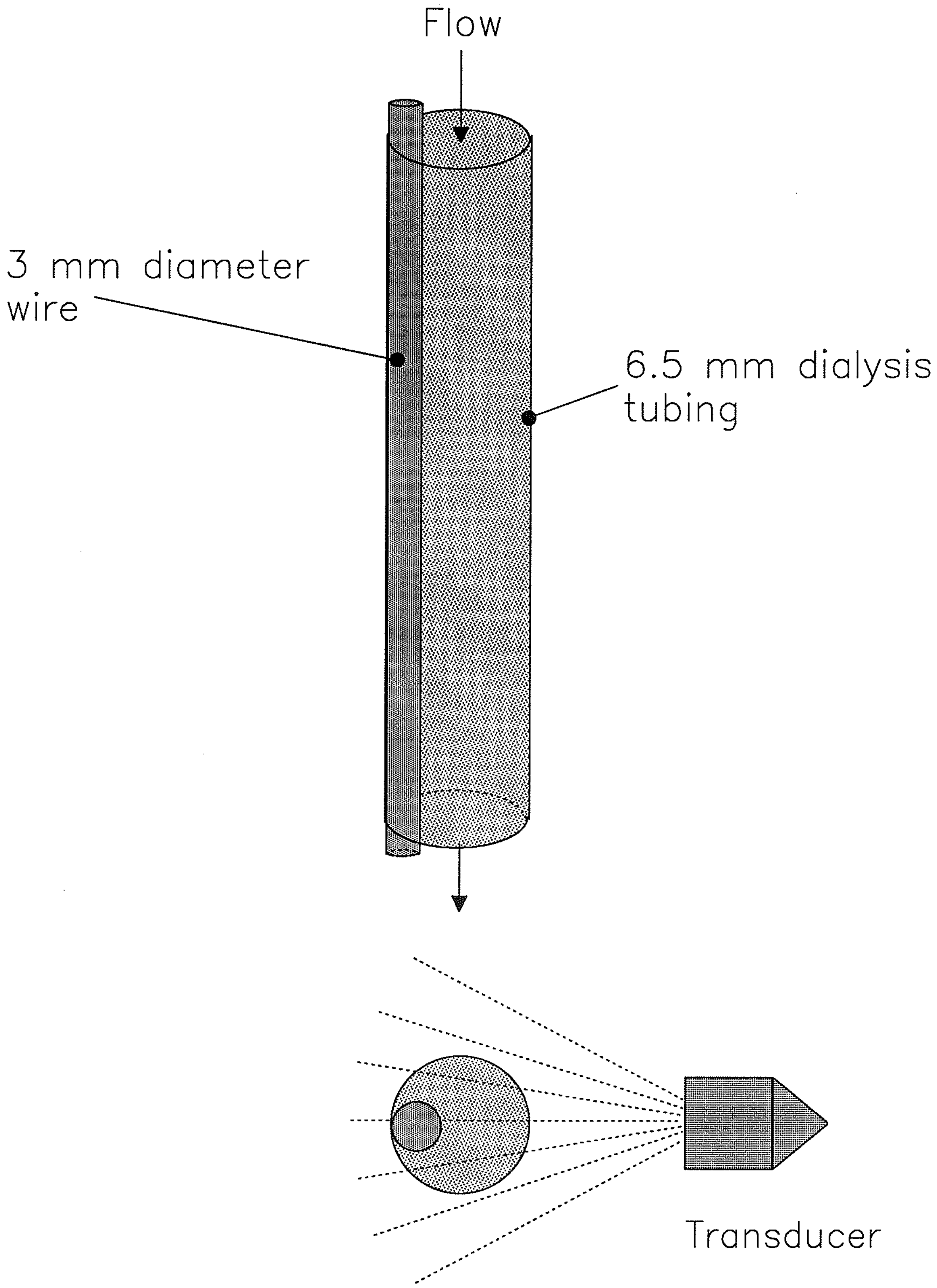


Figure 35 - Occlusion setup and transducer scanning configuration.

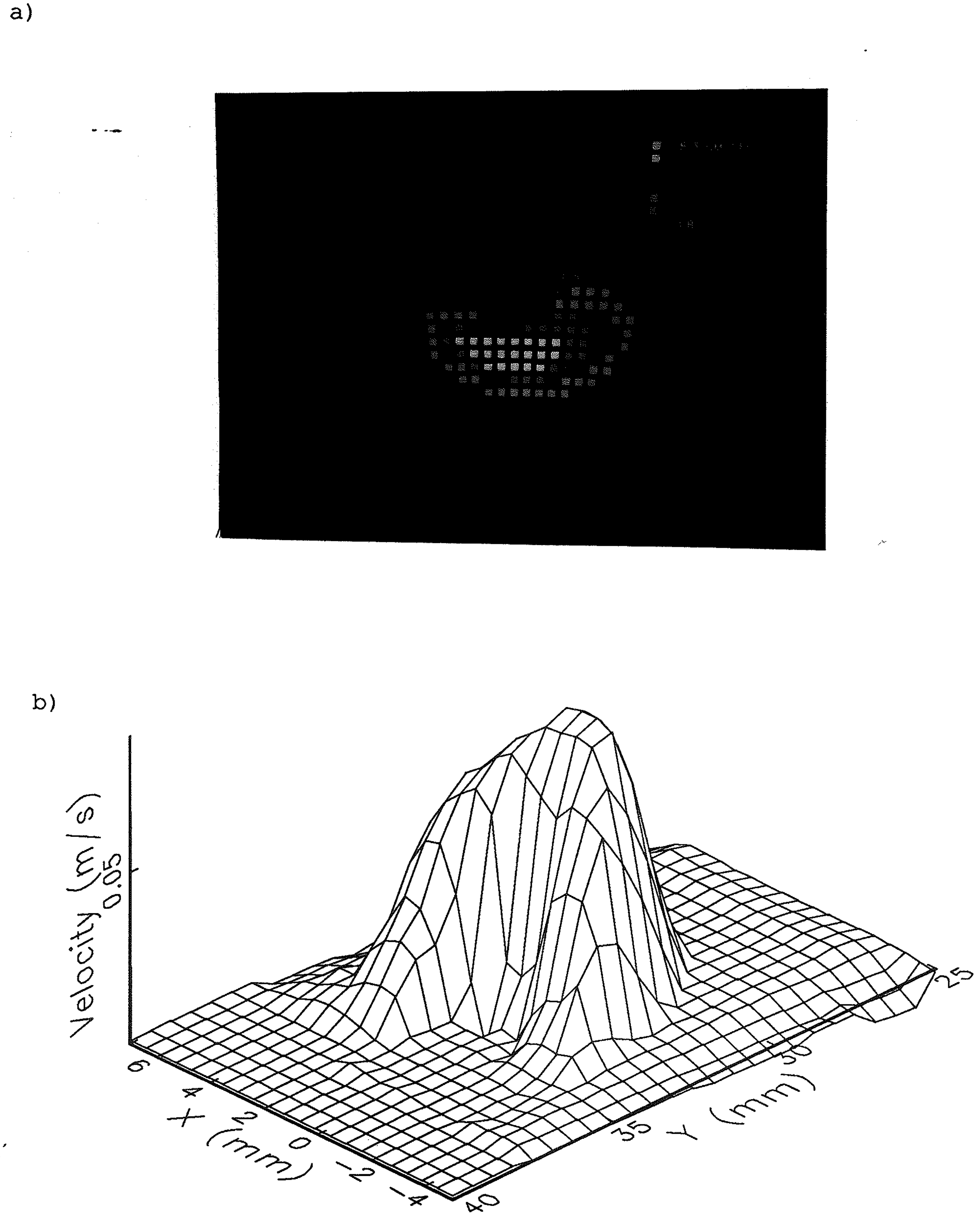


Figure 36 - a) Color-flow profile of occluded vessel.
b) Three-dimensional perspective of velocities around the occlusion.

vessel shape has changed. The 3-D plot in Figure 36b shows that the one-dimensional profiles remain somewhat parabolic as the vessel is scanned from left to right (grid lines running parallel to the y-axis). An analysis of the 3-D plot shows that the velocity gradient is greater from the occlusion to maximum velocity than from the front wall to maximum velocity (i.e., the velocity change is smoother and slower from the front wall to the maximum than from the occlusion to the maximum).

In the real-life situation an obstruction in a blood vessel may also produce turbulent flow. It was not possible to create an occlusion which will produce turbulent flow in the current blood flow phantom system. Future modifications to the blood flow phantom system should be able to produce turbulent flow so that the UTDC technique could be evaluated under those conditions.

5.2.2 Volumetric flow calculation by sum-of-differential flows

Since the flow through the occluded vessel is non-symmetric, the volumetric flow algorithms described in Chapter 3 could not be used to estimate the volumetric flow rate. One method for estimating the volumetric flow can be done by assuming the transducer is far away from the vessel and that the individual 1-D scans are approximately parallel to each other. With this approximation each data point can be assigned a differential rectangular area. The differential flow for a given data point area can be calculated by multiplying the data point velocity by the differential area as shown in Figure 37. The length of the differential rectangle chosen for the occlusion experiment is 0.6 mm, which is the length of the correlation window. The differential width was chosen to be the width of the vessel divided by the maximum number of 1-D scans intersecting the vessel, which is 0.38 mm. The volumetric flow was calculated by adding up all the individual differential flows Q_n for the n data points within the vessel walls. An experiment with a non-occluded vessel was performed to determine the measurement angle θ which was 52.5°. The error for the non-occluded vessel flow estimate using this method was -11.7%. The final error in the volumetric flow estimate for the occluded vessel was +24.5%.

At first it was thought that a major source of error in this estimate was due to the vessel not being far enough away from the transducer for the parallel 1-D scan assumption to be valid. The measurement configuration is cylindrical in nature and the data points are spaced closer at the front of the vessel than at the rear. There will be locations where differential areas overlap and there will also be areas of the vessel that are not covered by a rectangle, as in Figure 37.

A better differential area would be the sector of an annulus, as shown in Figure 38. This type of area will provide total coverage of the vessel and there will be no areas of overlap. The

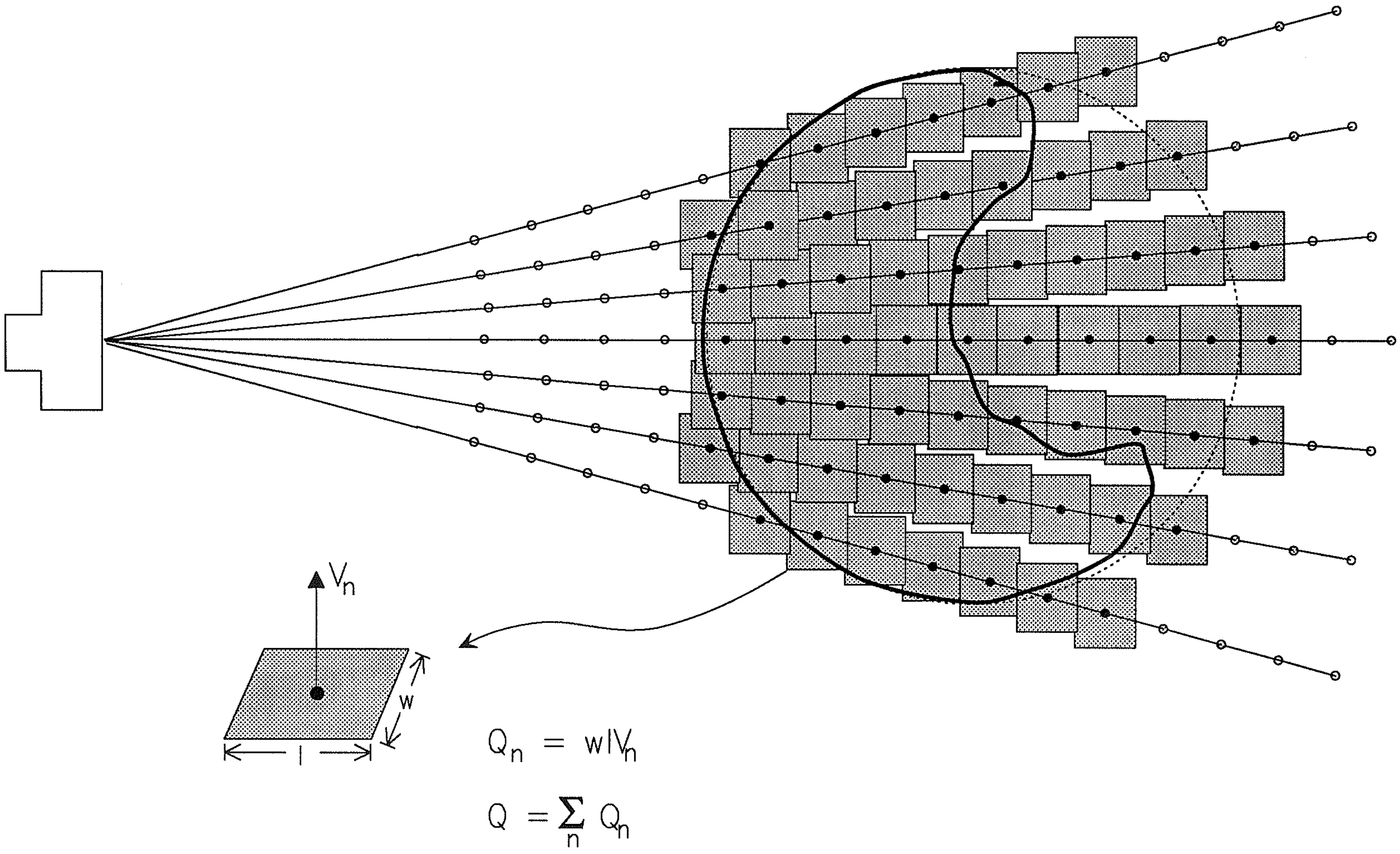


Figure 37 - Estimation of the volumetric flow by assigning each velocity data point V_n a rectangular area. The volumetric flow Q is estimated by summing all of the differential flows Q_n.

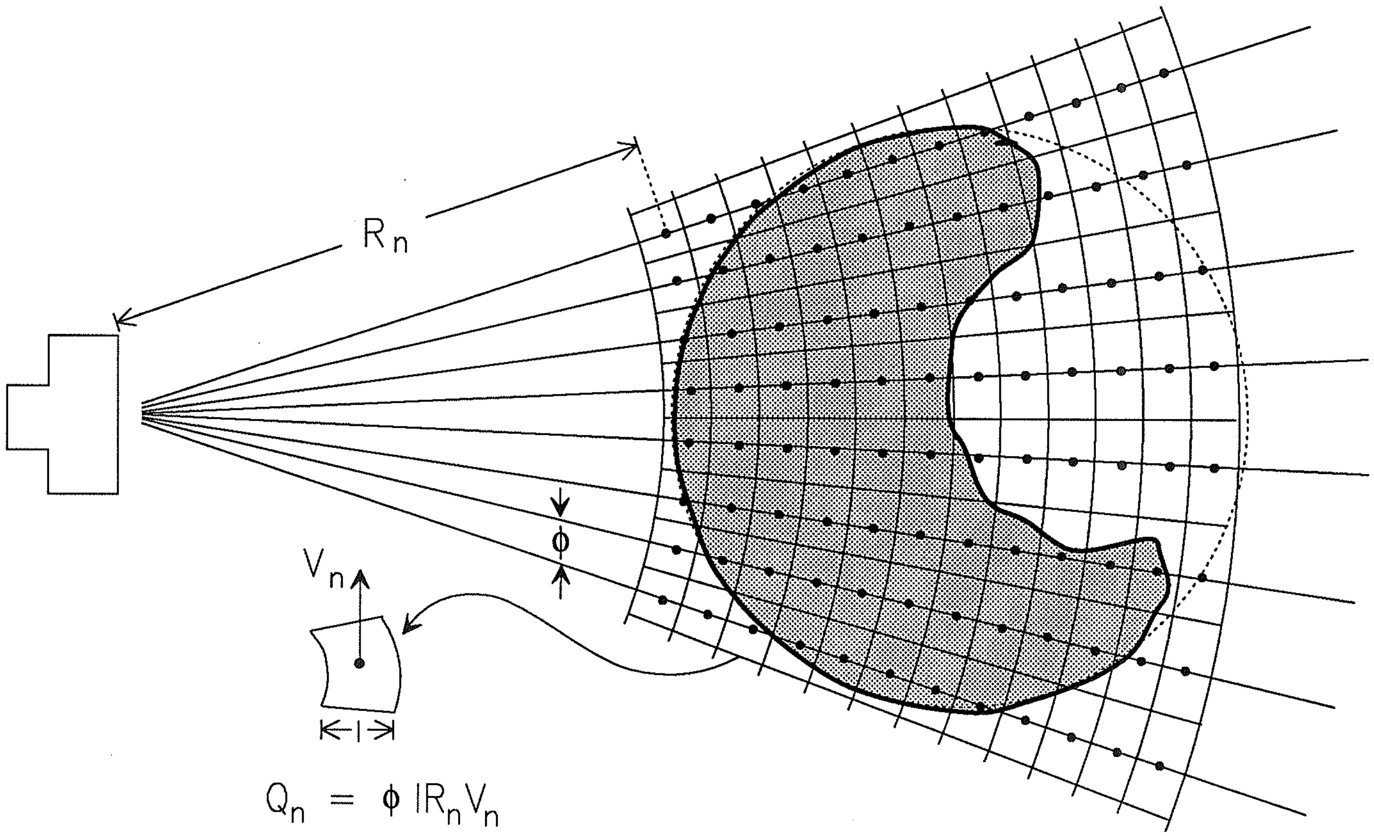


Figure 38 - Using polar coordinates and the sector of an annulus as the differential area will have no overlap and no missed areas of the vessel.

equation for the area of a sector of an annulus is

$$A_{sa} = \phi l R_n \qquad (42)$$

where ϕ is the angle between the 1-D scans, R_n is the distance from the transducer to the n_{th} data point, and l determined by the correlation length, 0.6 mm. The volumetric flows for both the non-occluded and the occluded vessels were recalculated with the sector-of-an-annulus differential area and are listed in Table 1:

Table 1: Error in flow estimates for rectangular and sector-of-annulus differential area (compared to hydrodynamic).

Experiment	Rectangular Error	Sector-of-Annulus Error
Normal	-11.7%	-2.2%
Occluded	+24.5%	+23.9%

This table shows that the flow estimate calculated by the sector-of-annulus differential area improved by 9.5% for the normal vessel but stayed approximately the same for the occluded vessel. This may be due to the fact the velocity gradients within the occluded vessel are greater than for the normal. For example, the velocities drop from the red and lavender levels to zero in the color flow plot of Figure 36a), while there are no such sudden velocity changes from the normal, fully developed laminar flow (Figure 34a)). The areas with high velocity near the border of the occlusion may overlap into the occlusion (where the flow rate is zero) thereby causing an over-estimation in the flow. This would occur for both the rectangular and sector-of-annulus differential areas. More axial and lateral resolution (smaller l, more 1-D scans with a smaller ϕ) may be required to accurately estimate flows that have high velocity gradients.

5.2.3 Multiple-vessel experiment

Another possibility existing in living systems is that of vessels located next to each other. The dialysis tubing section of the blood flow phantom was modified to accommodate two 6.5 mm vessels and is shown in Figure 39a). The transducer was oriented to the vessels as shown in Figure 39b). The color flow image and 3-D perspectives are shown in Figure 40. The cutoff for the color flow image was set at zero and shows the sector-scan nature of the measurement. The volumetric flow and measurement angle for each vessel were calculated separately using the 2-D algorithms described in [10], and the total flow through both was compared to the total hydrodynamic flow (it was not possible to measure the hydrodynamic flow in each individual vessel without major modification to the blood flow phantom). The flow did not appear to split equally between the vessels (384 ml/min and 298 ml/min) but the sum of the flows agreed to within 13% of the total hydrodynamic flow. There is no reason why there should be any additional error introduced by two vessels instead of one in this particular transducer configuration since each 1-D scan passes

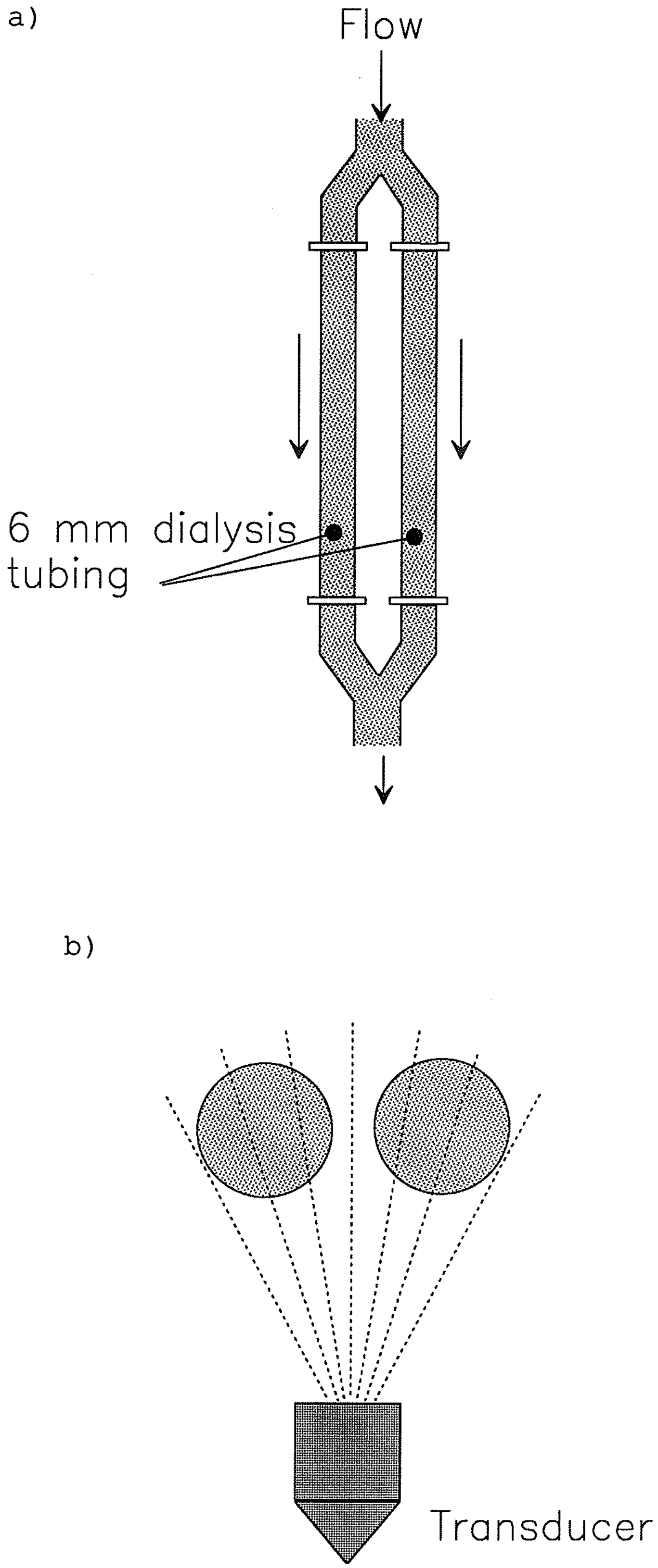


Figure 39 - a) Modification of blood flow phantom to accommodate two vessels. b) Transducer scanning configuration.

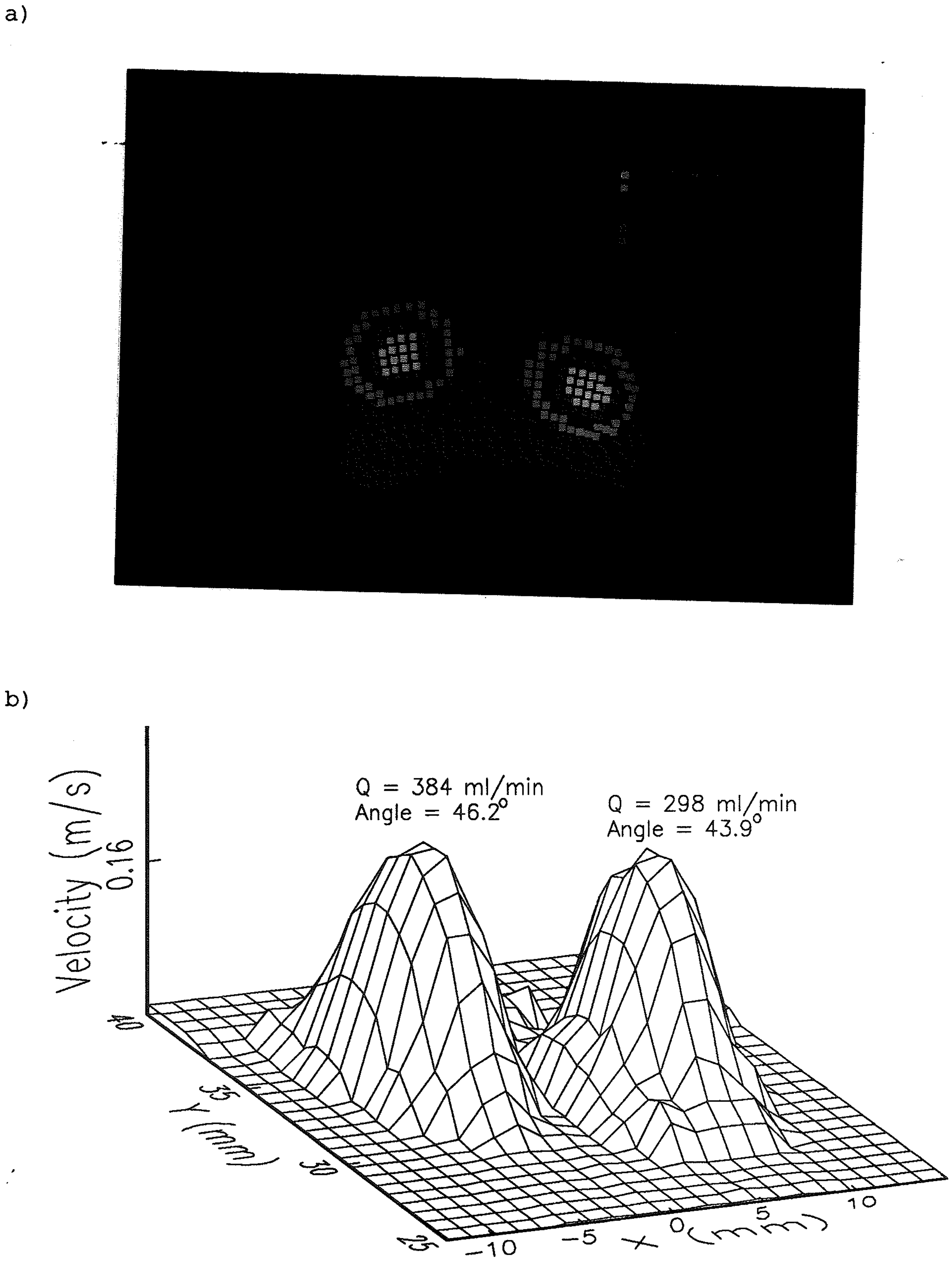


Figure 40 - a) Color-flow plot of flow velocities in two vessels. b) Three-dimensional perspective of flow through the vessels.

through only one vessel. A more interesting experiment would be to position one vessel behind the other (with respect to the transducer) so that a 1-D scan would pass through both vessels. Experiments could then be performed to see if the existence of the front vessel would affect the flow estimate in the rear vessel or vice versa. With this transducer positioning system (see [9]), it was not possible to orient the transducer to the vessels in this manner. However, with the real-time system described in Chapter 8, there are no restrictions on the placement of the transducer and this type of experiment was performed. The details and results of two-vessel experiments with one behind the other are described in Chapter 8.

5.3 *In-vivo* Experiment

Attempts were also made to measure in-vivo flow in a human subject. The flow measurement in the carotid artery was attempted by removing the transducer from the transducer positioning system and directing it in the general direction of the carotid artery (through an acoustical standoff consisting of a latex glove filled with distilled water), as shown in Figure 41. Artery location was attempted by observing the ultrasound RF echoes on an oscilloscope and looking for pulsations in the signal due to the subject's heartbeat. The subject was the author who was also doing the measurements, thus the subject's heartbeat is known and correlatable to pulsations in the RF signal, if observable. A

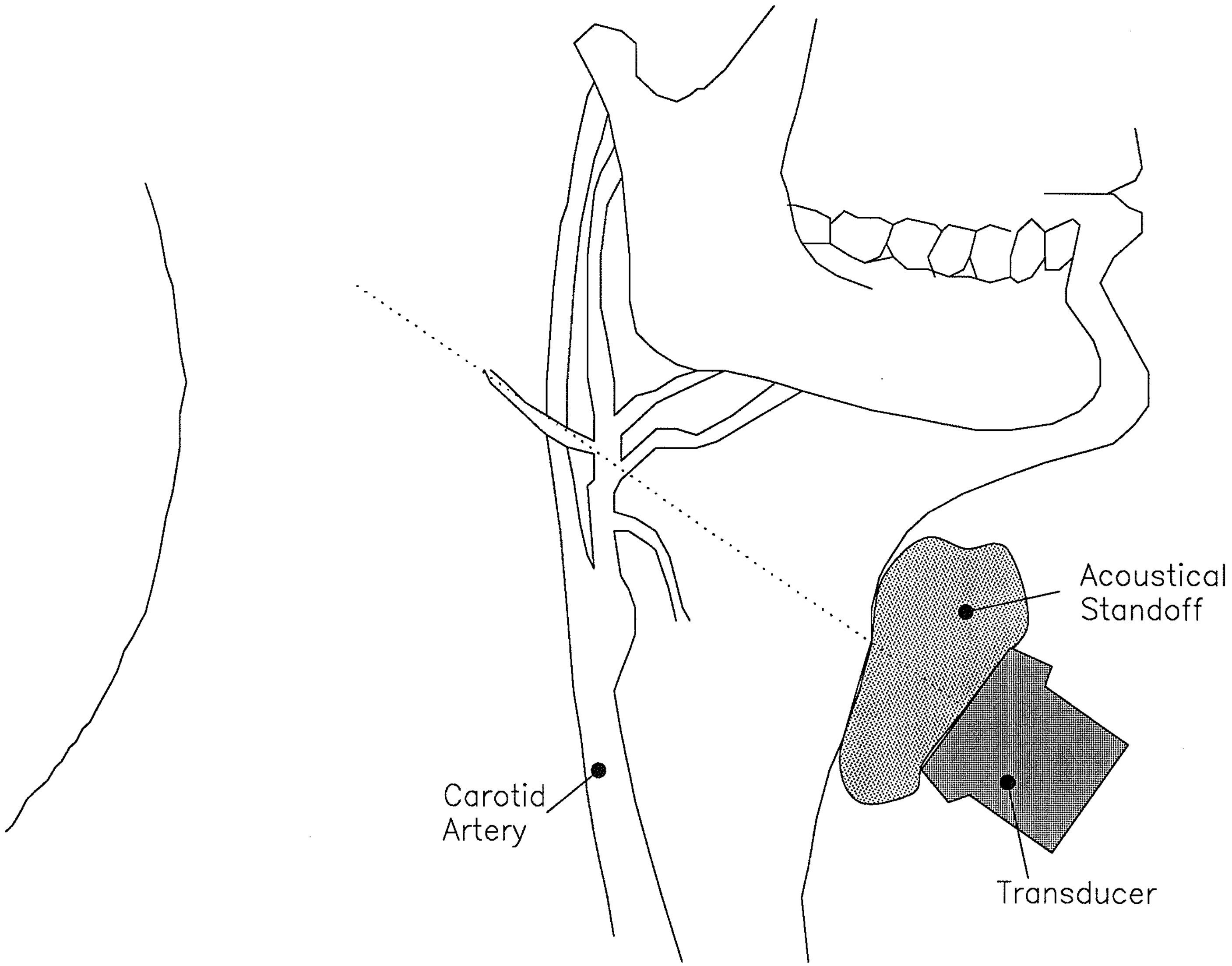


Figure 41 - Carotid artery measurement using previous system.

large number of 1-D measurements (approximately 35) were performed in the hopes of intersecting the carotid artery. Eventually the artery was intersected with the result shown in Figure 42. It appears that two vessels may have been intersected, which is a reasonable assumption since there are numerous bifurcations in this artery. This is not a quantitative result since it is not known what the measurement angle was, which part of the vessel was intersected, or what part of the cardiac cycle the subject was in. It does, however, indicate that the use of the UTDC technique *in-vivo* is feasible. It also demonstrated the need for an ultrasound imaging system, since it is extremely difficult to locate a vessel with the above described technique.

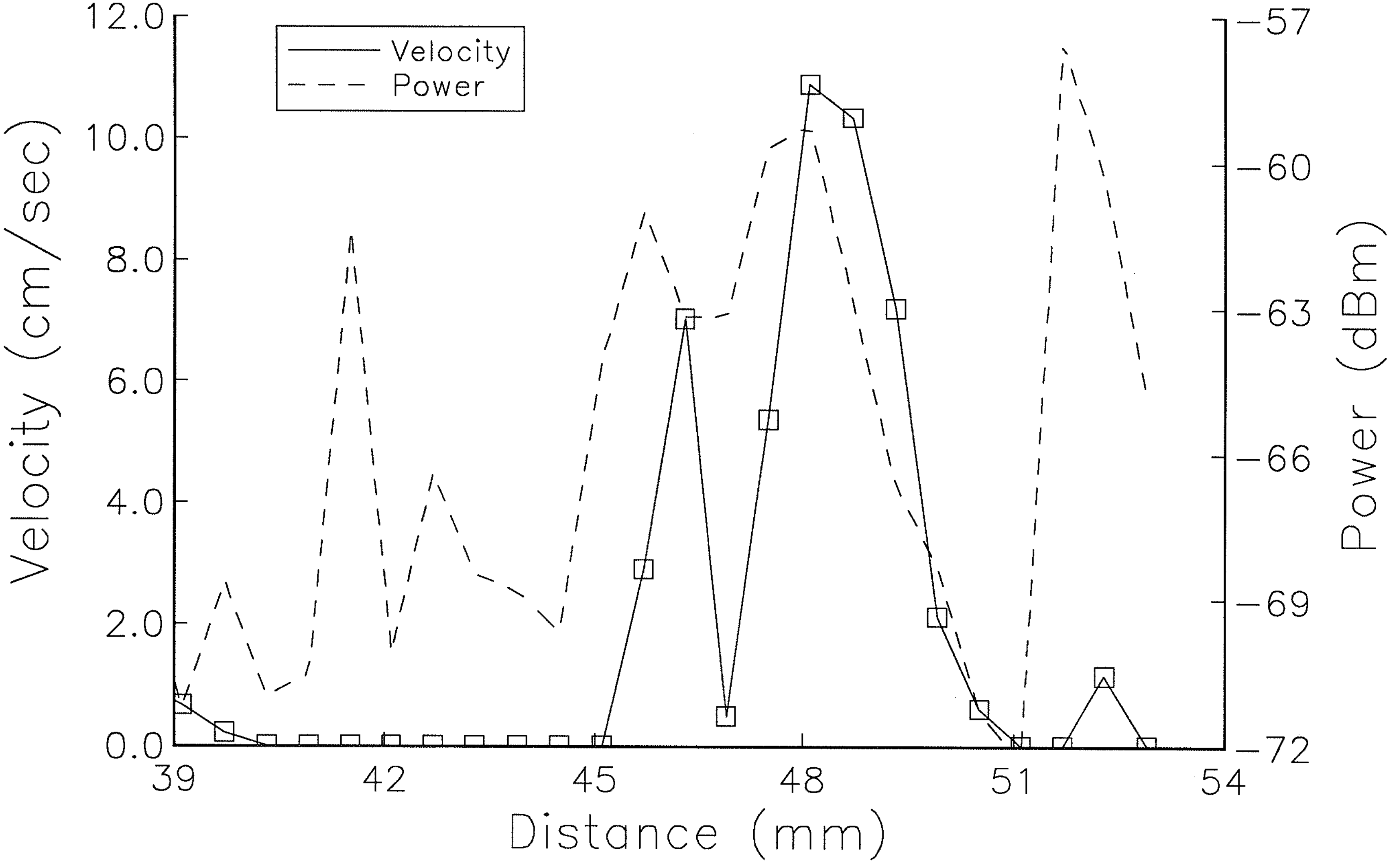


Figure 42 - One-dimensional velocity profile of flow within a human carotid. The reflected power is also shown.

CHAPTER 6

PULSATILE FLOW MEASUREMENTS

Pulsatile flow was generated in the blood flow phantom system over the range of 0 to 150 beats/minute. The flow velocity profile was measured across the vessel using the ultrasound time-domain correlation technique. The transducer measurement angle, the vessel wall diameter, and the total volumetric flow were calculated from the flow velocity profiles.

6.1 Data Acquisition

Figure 43 illustrates the pulsatile flow measurement system. Ultrasonic pulses were generated with a 5-MHz transducer and were digitized and stored at 50 MHz. The ultrasound data acquisition and calculation of time shifts were made by the ultrasound data acquisition system (UDAS) described in [31]. The UDAS system has been interfaced to a COMPAQ 386/20 computer which controls the UDAS and calculates the flow velocity profiles from the time shifts calculated by the UDAS.

The SWF-5RD electromagnetic flowmeter can display the instantaneous volumetric flow on an LED display on the front panel and also has an analog output for a plotter. The plotter output

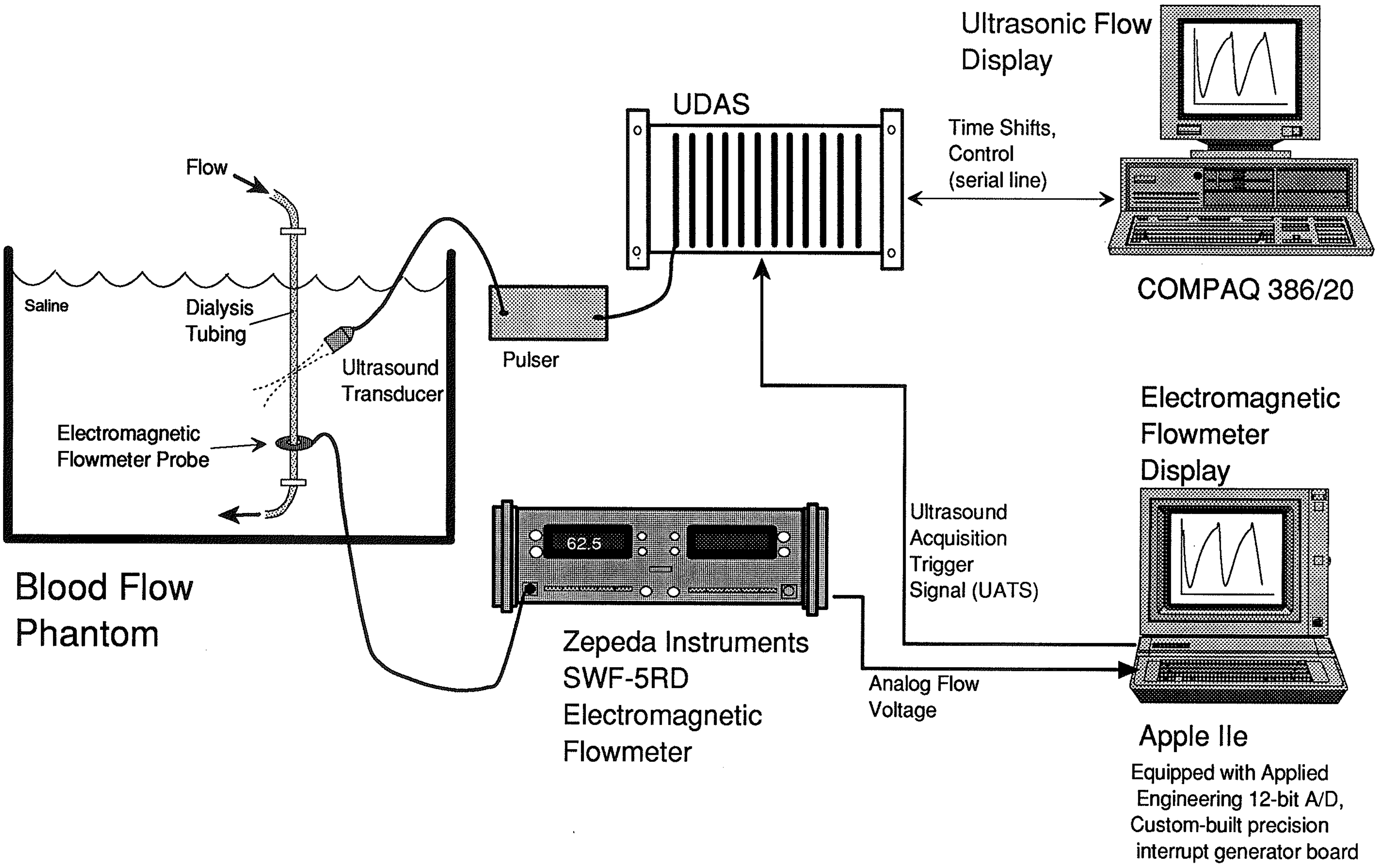

Ultrasonic Flow Display
UDAS
Time Shifts, Control (serial line)
COMPAQ 386/20
Flow
Saline
Dialysis Tubing
Ultrasound Transducer
Pulser
Electromagnetic Flowmeter Probe
Electromagnetic Flowmeter Display
62.5
Ultrasound Acquisition Trigger Signal (UATS)
Blood Flow Phantom
Zepeda Instruments SWF-5RD Electromagnetic Flowmeter
Analog Flow Voltage
Apple IIe
Equipped with Applied Engineering 12-bit A/D, Custom-built precision interrupt generator board

Figure 43 - Pulsatile flow measurement system.

is digitized by an Apple IIe computer equipped with a 12-bit A/D board and a custom-built precision interrupt generator board (details in Appendix C). The Apple IIe acts as a digital scope and displays the pulsatile volumetric flow rate vs. time for the electromagnetic flowmeter. The synchronization of data acquisition between the Apple IIe and the COMPAQ 386 is accomplished via an ultrasound acquisition trigger signal (UATS) provided by the Apple computer. The generation of the UATS is controlled by the digital scope program written for the Apple. In normal use, the Apple is continuously displaying the volumetric flow from the electromagnetic flowmeter. A trigger level can be set such that when the volumetric flow reaches a certain level the UATS is sent to UDAS, which will cause UDAS to digitize and store ultrasound echoes and the COMPAQ will calculate the flow rate from the ultrasonic data. The Apple will also store the electromagnetic flow data to a floppy disk at the time when the UATS is generated. In this manner the ultrasound data acquisition and electromagnetic data acquisition are synchronized to the same point in time.

For pulsatile flow experiments the UDAS is capable of two types of data acquisition. It can perform a zero-dimensional (point) velocity measurement at a very small range cell width (0.6 mm) at a programmable distance r from the transducer for a comparatively long time (1 sec) to provide a velocity vs. time profile at that cell location, or it can perform a one-dimensional velocity measurement across a 15 mm range very quickly (compared to the pulsatile flow rate) to produce a one-dimensional velocity vs. distance profile at that instant in time. To increase calculation speed, the point velocity measurement uses only ten echoes and the time shifts calculated from the different delta values are weighted equally (the weighted averaging method is not used). The one-dimensional measurement uses 384 echoes and the weighted averaging method is used to calculate the velocity.

6.2 Flow Determination

The determination of pulsatile flow consists of two steps. The first step is to perform a two-dimensional constant flow experiment with the pulsatile pump off in order to determine the transducer measurement angle θ. This step is necessary because the UDAS system was designed to measure constant flow and is not fast enough to process a two-dimensional measurement for pulsatile flow. After the two-dimensional measurement, a number of one-dimensional measurements are made (still at constant flow) at small increments of the transducer scan angle α (as defined in Figure 29). The scan angle α with the largest velocity profile corresponds to the transducer beam aimed directly through the center of the vessel. All of the pulsatile one-dimensional velocity vs. distance profiles are measured at this angle α_{max}. A number of different velocity vs. time measurements are also made at different ranges r from the transducer at the angle α_{max}. The velocity vs. time profile for constant flow will be flat (at a given r location) and the level will vary as r is changed due to the parabolic flow shape within the vessel. The value of r where the level is maximum corresponds to the location of the center of the vessel. All of the pulsatile velocity vs. time profiles are made at this range r_{max} at the angle α_{max}.

Once the measurement angle θ is known and the transducer is oriented at α_{max} the pulsatile pump is activated. The vessel diameter D and the volumetric flow rate Q can then be estimated from the one-dimensional velocity profiles. The diameter of the vessel will be given by

$$D = (d_r - d_f) \sin(\theta) \qquad (43)$$

where d_r is the location of the rear wall of the vessel and d_f is the location of the front wall. The wall locations are considered the points in the velocity vs. distance profile where the velocity becomes zero. The volumetric flow rate Q can be determined for fully developed laminar flow from the equation

$$Q = 0.5\ V_{max}\ A \cos(\theta) \qquad (44)$$

where V_{max} is the maximum velocity at the center of the vessel and A is the cross-sectional area of the vessel $\pi(D/2)^2$. The volumetric flow was determined using Equation (44) at the maximum and minimum flow peaks of the pulsatile flow waveform. UTDC one-dimensional measurements were made at the peaks by setting the UATS trigger level on the Apple computer at either the maximum or minimum flow peak on the electromagnetic flow display. The peak-to-peak flow was calculated by subtracting the minimum flow peak from the maximum and the average flow by adding the minimum and maximum and dividing by two.

6.3 Results

A total of 25 experiments ranging from 50 to 150 beats/minute (b/min) were made with flow rates ranging from approximately 50 to 1100 ml/min. Constant flow experiments were also performed throughout the range of volumetric flows to ensure the flow was laminar. The flow was found to be fully developed and laminar to at least 1000 ml/min. Figure 44 shows the elliptic curve fit to data at a constant flow rate of 462 ml/min. The hydrodynamic flow was measured to be 449 ml/min, a 3% error. The measurement angle θ was set at 45° and estimated from Figure 44 to be 51.9°.

6.3.1 Volumetric flow results

The pulsatile volumetric flow was calculated from the one-dimensional scans through the center of the vessel. Figure 45 shows that the velocity profile at the maximum and minimum flow peaks for pulsatile flow is approximately parabolic and also shows that the vessel wall diameter expands to 8 mm at the maximum flow peak and contracts to 4.5 mm at the minimum flow peak.

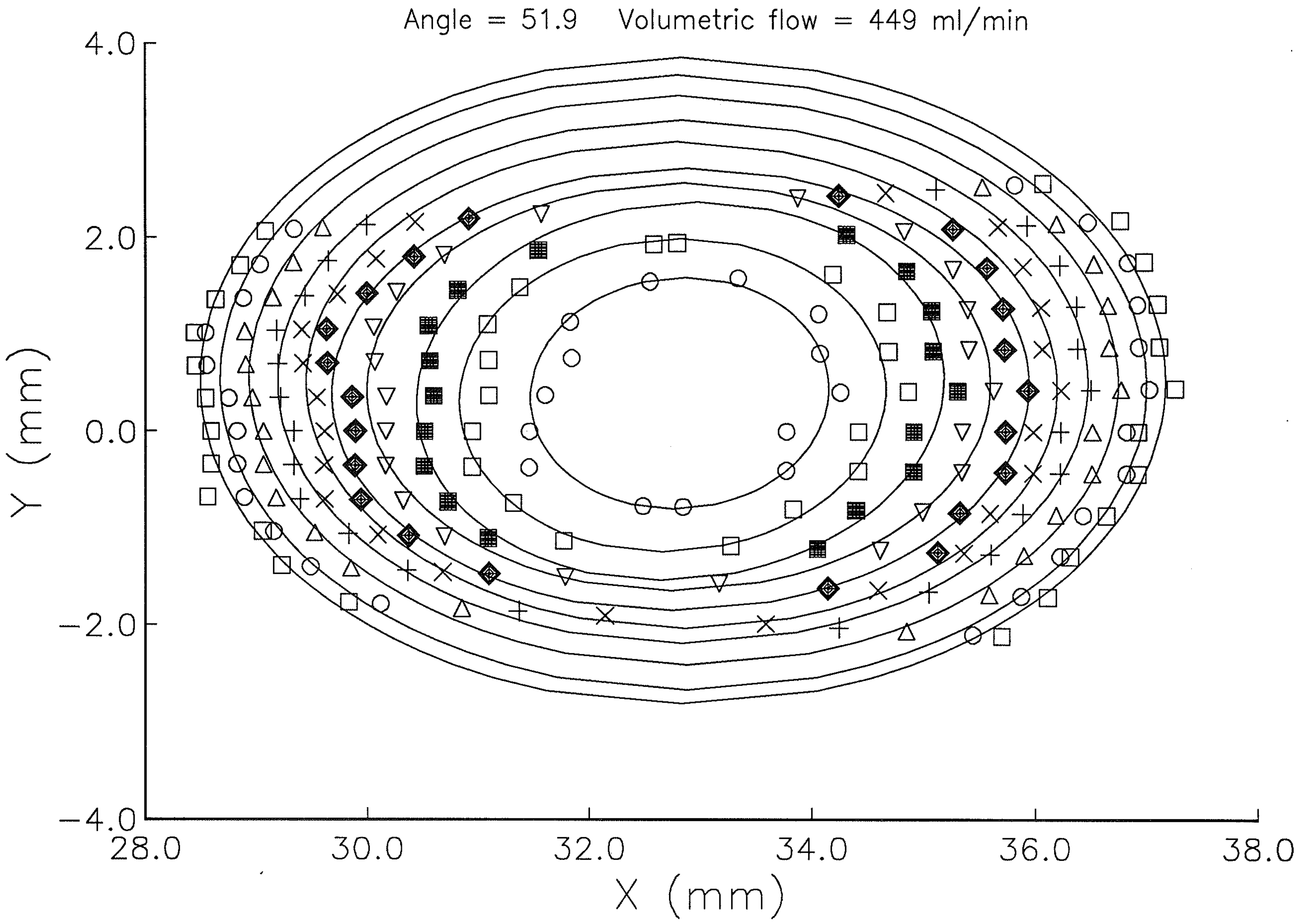


Figure 44 - Constant velocity ellipses used to calculate the transducer measurement angle for the pulsatile experiments.

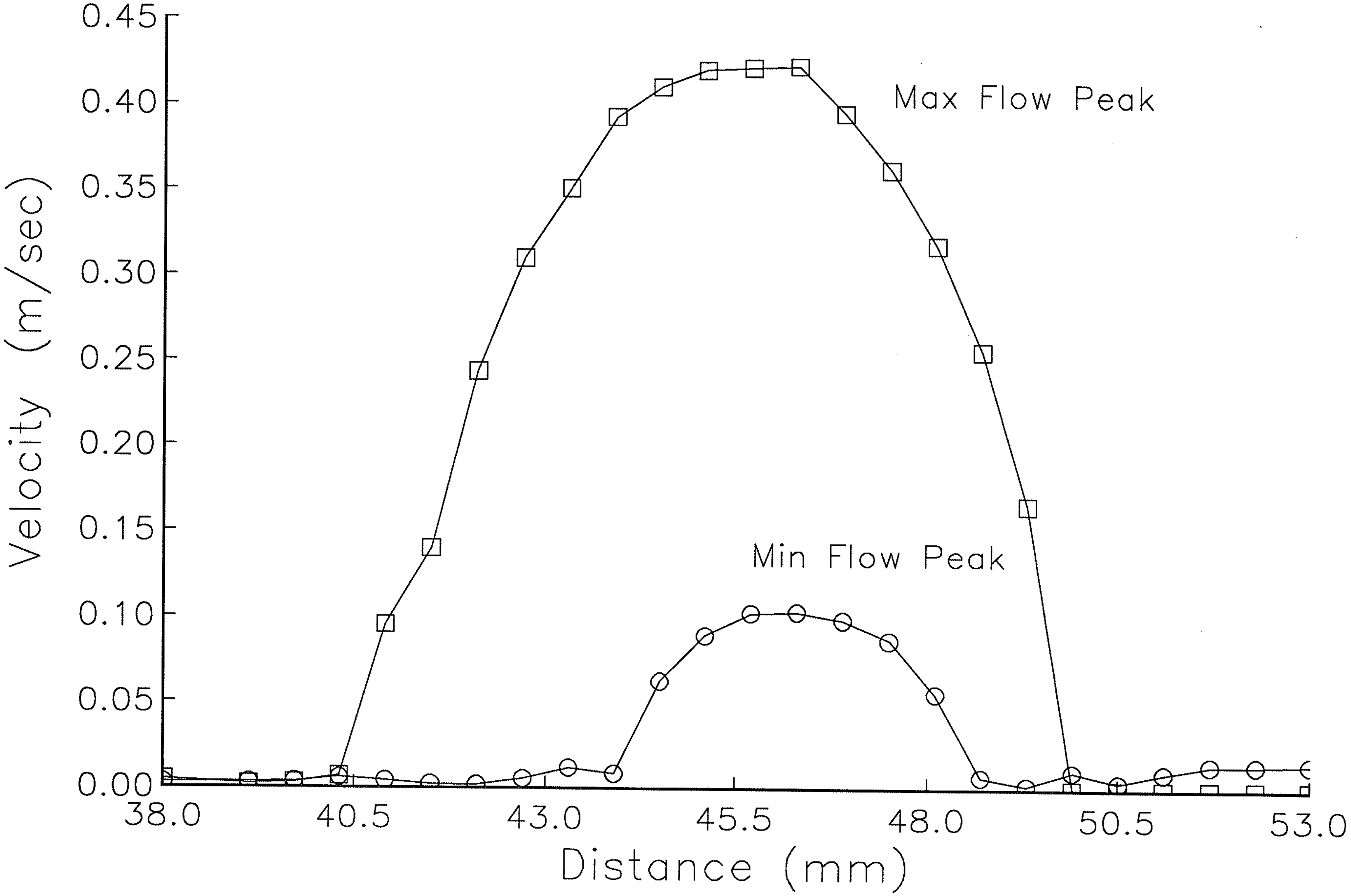


Figure 45 - Velocity profiles of the minimum and maximum flow peaks at 120 b/min. The profiles are approximately parabolic and show how the vessel wall diameter changes.

Table 2 lists the results of all the pulsatile flow measurements made. The vessel wall diameter, the maximum and minimum flow peaks, the peak-to peak and average flows calculated from the maximum and minimum peaks, and the error referenced to the electromagnetic flowmeter are shown.

6.3.2 Flow vs. time

Figures 46a) and 46b) show flow vs. time for 86, 100, 120, and 150 b/min as measured by the electromagnetic flowmeter and as measured by the UTDC technique. The electromagnetic flowmeter is only capable of measuring volumetric flow through the vessel; hence, the total volume flow vs. time is plotted in Figure 46a), while the velocity at the center of the vessel is plotted in Figure 46b). The volumetric flow rates have also been shown at the maximum and minimum flow peaks.

6.4 Discussion of Volumetric Flow Results

A summary of the UTDC technique measurement errors as compared to those for the electromagnetic flowmeter measurement is plotted in Figure 47. The error between the ultrasonic and electromagnetic flowmeter measurements was within about 18% for both the peak-to-peak and average flow up to 125 b/min (see Table 2). The average flow error appears to increase to 24% at 150 b/min. This is mainly due to the error in the measurement of the

Table 2: Results of pulsatile flow experiments.

Data	Frequency		Walls		Max Peak			Min Peak			Pk-Pk Flow			Average Flow		
Set			(mm dia)		(ml/min)		Error	(ml/min)		Error	(ml/min)		Error	(ml/min)		Error
	Hz	BPM	Max	Min	UTDC	EMF	(%)	UTDC	EMF	(%)	UTDC	EMF	(%)	UTDC	EMF	(%)
C120	.83	50	7.06	5.88	419	426	- 1.6	122	134	- 8.9	297	292	- 1.5	270	280	+ 3.5
W120			6.68	5.48	396	423	- 6.3	131	147	-10.8	264	275	+ 3.8	263	285	+ 7.6
P100	1.00	60	7.19	5.12	405	423	- 4.2	87	71	+22.5	318	352	+ 9.9	246	247	- 1.8
B100			7.18	5.76	423	427	- 0.9	91	63	+44.4	331	363	+ 8.8	256	245	- 4.5
C100			7.09	4.60	429	445	- 3.5	63	65	- 3.1	366	380	+ 3.4	246	254	+ 3.4
W100			6.76	5.22	384	444	-15.6	100	100	0.0	283	343	+17.3	272	241	+11.0
C90	1.11	67	7.31	4.97	428	445	- 3.8	36	28	+28.6	374	417	+10.0	236	240	- 1.8
B80	1.25	75	7.56	5.60	531	511	+ 3.9	67	0		462	510	+ 9.5	299	255	-17.0
D80			7.10	4.60	487	537	- 9.3	42	0		446	537	+16.8	268	265	+ 1.2
W80			6.65	5.22	425	478	-11.1	87	67	+40.3	338	415	+18.5	270	255	+ 5.3
F80			7.48	4.65	520	495	+ 5.5	98	62	+58.1	421	432	+ 2.4	278	309	-10.9
G80			7.06	4.50	449	460	- 2.3	73	53	+37.8	375	407	+ 7.7	260	256	- 1.5
I80			7.36	6.37	874	818	+ 6.8	450	433	+ 6.3	424	385	+10.1	662	625	+ 5.9
F70	1.43	86	7.65	4.52	576	505	+14.1	73	47	+55.3	503	457	- 9.9	324	276	-17.6
G70			7.68	4.67	599	556	+ 7.7	94	82	+14.6	505	474	- 6.6	346	319	- 8.6
I70			7.25	5.88	801	797	+ 0.5	321	352	- 8.8	480	445	- 7.9	560	574	+ 1.1
X70			7.30	5.22	907	902	+ 0.5	84	0		822	901	+ 8.7	495	450	- 9.9
F60	1.67	100	7.70	4.52	824	759	+ 8.5	100	47	+ 112	728	711	- 2.3	464	403	-15.2
G60			7.23	4.78	667	674	- 1.0	115	70	+64.8	551	604	+ 8.6	393	372	- 5.2
I60			7.48	5.18	817	793	+ 3.0	192	196	- 2.1	625	595	- 4.9	504	494	- 1.8
F50	2.00	120	8.41	4.51	1022	957	+ 6.7	77	0		946	956	+ 1.2	549	478	-14.8
G50			7.83	4.66	962	870	+10.4	133	97	+37.1	828	773	- 7.2	547	484	-13.1
I50			7.87	5.29	992	982	+ 1.0	194	113	+71.7	798	868	+ 8.1	593	547	- 8.3
H40	2.50	150	8.61	4.83	1047	901	+16.2	98	67	+46.3	949	832	-14.0	572	483	-18.3
I40			8.64	4.59	1095	959	+14.1	99	0		995	959	- 3.8	597	479	-24.4

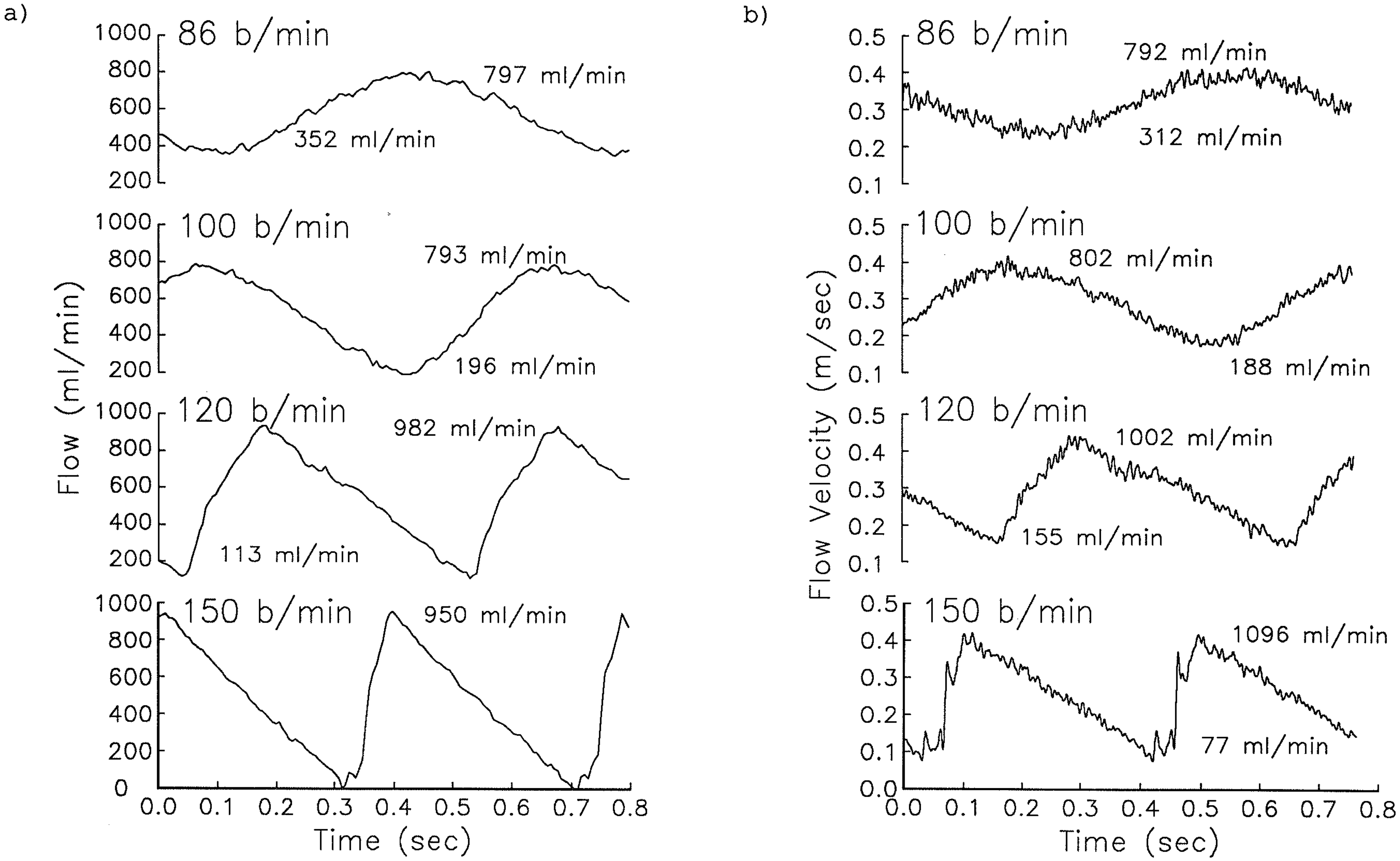


Figure 46 - a) Pulsatile flow measured by the electromagnetic flowmeter. The total volumetric flow in ml/min is plotted. b) Pulsatile flow measured by the UTDC technique. The velocity in m/sec of a 0.6 mm range cell at the center of the vessel is plotted.

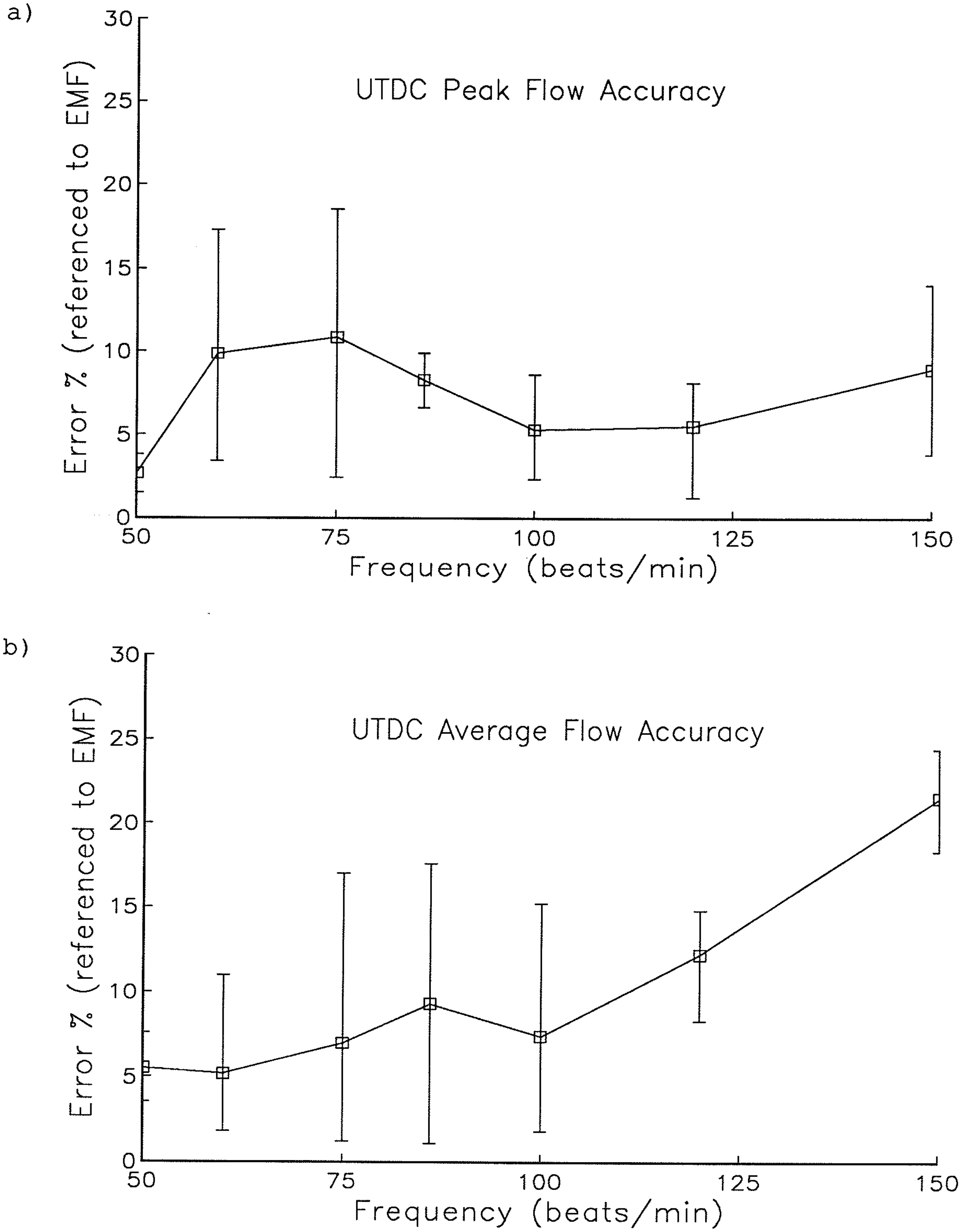


Figure 47 - a) Error of the UTDC peak-to-peak flow measurement as compared to that for the EMF. b) Error of the UTDC average flow measurements. These graphs represent a composite of all the pulsatile experiments performed. The error bars show the maximum and minimum errors encountered for a given frequency.

minimum flow peak, which has a much larger error than the maximum flow peak. The errors in the maximum flow measurements are within 15%; however, the minimum flow errors vary widely and are almost always positive. There are three possible sources of error to account for a greater error at minimum flow compared to maximum flow conditions.

A major source of error appears to be due to the electromagnetic flowmeter probe. The probe has an internal diameter of 6 mm; the normal diameter of the dialysis tubing is approximately 6.5 mm. Since the dialysis tubing is elastic, it changes diameter with the changing flow rate. It expands to a maximum diameter at the maximum flow peak and contracts to a minimum at the minimum flow peak. The vessel wall diameter vs. frequency for a set of experiments has been plotted in Figure 48 to illustrate how the vessel diameter changes with frequency. From Figure 48, as well as Table 2, it is apparent that for most of the experiments the vessel wall diameter is under 6 mm, which causes the electromagnetic flowmeter measurement to be less reliable. In extreme cases, the probe totally lost contact with the vessel, which caused a zero measured flow rate by the electromagnetic flowmeter. This is demonstrated in Figure 46a) for 150 b/min where the electromagnetic flowmeter indicates a minimum flow rate of zero while the UTDC technique measurement indicates a minimum flow of 77 ml/min. This is due to the fact that the vessel walls have lost contact with the electromagnetic

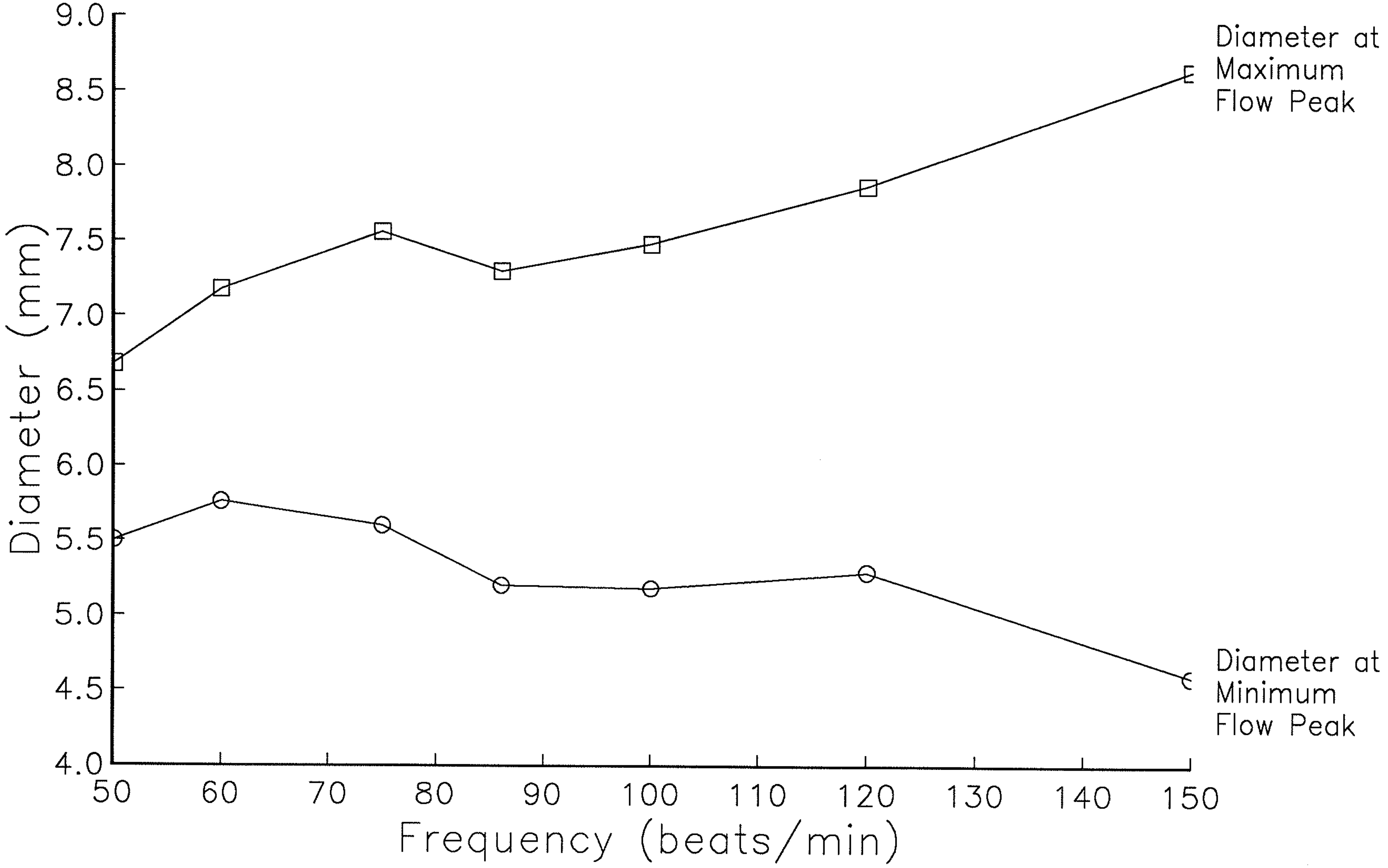


Figure 48 - Vessel wall diameter vs. frequency of pulsatile flow as measured by ultrasound time-domain correlation.

flowmeter probe, which caused a large error of 24% in the average flow measurement of I40.

A second source of error is that the minimum flow rates were near the lower limit of the dynamic range of the electromagnetic flowmeter. The maximum flow rates measured were 1100 ml/min; on this scale, measurements under 100 ml/min are less accurate.

This also may be true of the ultrasonic measurement. As a third source of error, the precision of the time-domain correlation technique is, in part, dependent on the pulse repetition period. For time-varying flow, faster flow velocities require a smaller period (higher PRF) while slower velocities require a larger period (lower PRF). For pulsatile experiments, the UTDC technique calculates the time shift from 10 consecutive echoes (45 echo pairs), which is an effective period range of T to 9T. The fundamental period T was set such that the maximum flow velocities were most accurately measured; it is possible that the minimum flow velocities were slower than could be accurately measured at 9T. To remedy this, more than 10 echoes may have to be used to produce a larger dynamic range.

In general, the error in the minimum flow peak has a much smaller effect on the overall error than at the maximum flow peak, since its value is small compared with the value for the maximum flow peak. Hence, the overall error for the pulsatile flow experiments was within about 18% (except for experiment I40).

6.5 Discussion of Flow vs. Time Results

A number of important observations can be made from the zero-dimensional (point) flow vs. time plots in Figure 46. One is that the shapes of the waveforms as measured by UTDC match very well those measured by the electromagnetic flowmeter. At first glance, the UTDC measurements appear more "noisy" than the electromagnetic flowmeter measurements. One reason for this is that the UTDC measurement is observing a very small range cell at the center of the vessel, while the electromagnetic flowmeter is observing the average of the flow velocities across the entire vessel's cross-sectional area. The averaged waveform will naturally look smoother than the non-averaged waveform. The point measurement also uses a very limited number of echoes and very little averaging takes place, which also contributes to the roughness of the ultrasound measurement.

Another interesting observation is that the flow vs. time waveform changes from a nice sinusoid at 86 b/min to a ramp function at 150 b/min. For a simple system, the displacement of fluid by the syringe in the pulsatile pump can be described as

$$d = A \sin(\omega t) \qquad (45)$$

where d is the displacement, ω is the fundamental frequency, and A is the maximum syringe displacement. The fluid velocity would be the derivative of the displacement, or

$$v = A\ \omega\ \cos(\omega t) \tag{46}$$

which indicates the amplitude should increase linearly with frequency. For low frequencies, this model is valid. However, at higher frequencies, the effects of the elasticity of the vessel walls as well as inertia and viscosity of the liquid become significant [32]. This has the effect of generating higher amplitude harmonics as indicated by the flow waveform changing from a sinusoid to ramplike in nature.

6.6 Conclusion

Previous research has shown that the UTDC technique can measure continuous volumetric flow in a circular vessel with an accuracy of about 15%. The work reported in this chapter has shown that it can measure with similar accuracy pulsatile volumetric flow in the frequency and flow ranges found in blood vessels.

CHAPTER 7

TEMPORARY SYSTEM

This chapter describes the temporary system. It was used for experiments while the real-time system was under construction. It is capable of imaging the region of interest where the velocities are to be measured and performing one-dimensional velocity measurements. The velocities are calculated using only twenty echoes, as opposed to 384 for the previous system.

7.1 Temporary System Setup

Figure 49 shows the setup of the temporary system. It consists of an ATL MK500 ultrasound imaging system (donated by ATL) interfaced to the COMPAQ 386 computer. The COMPAQ is equipped with a 40 MHz 8-bit A/D board (WAAG II, Markenrich Corporation). The ATL MK500 consists of two units, a pulsed Doppler flow analyzer and an 860C imager. The RF signal was extracted from the 860C imager and digitized. The /Depth_Gate signal was extracted from the pulsed Doppler flow analyzer and used to trigger the WAAG II board. The modifications to the ATL MK500 system are shown in Appendix D.

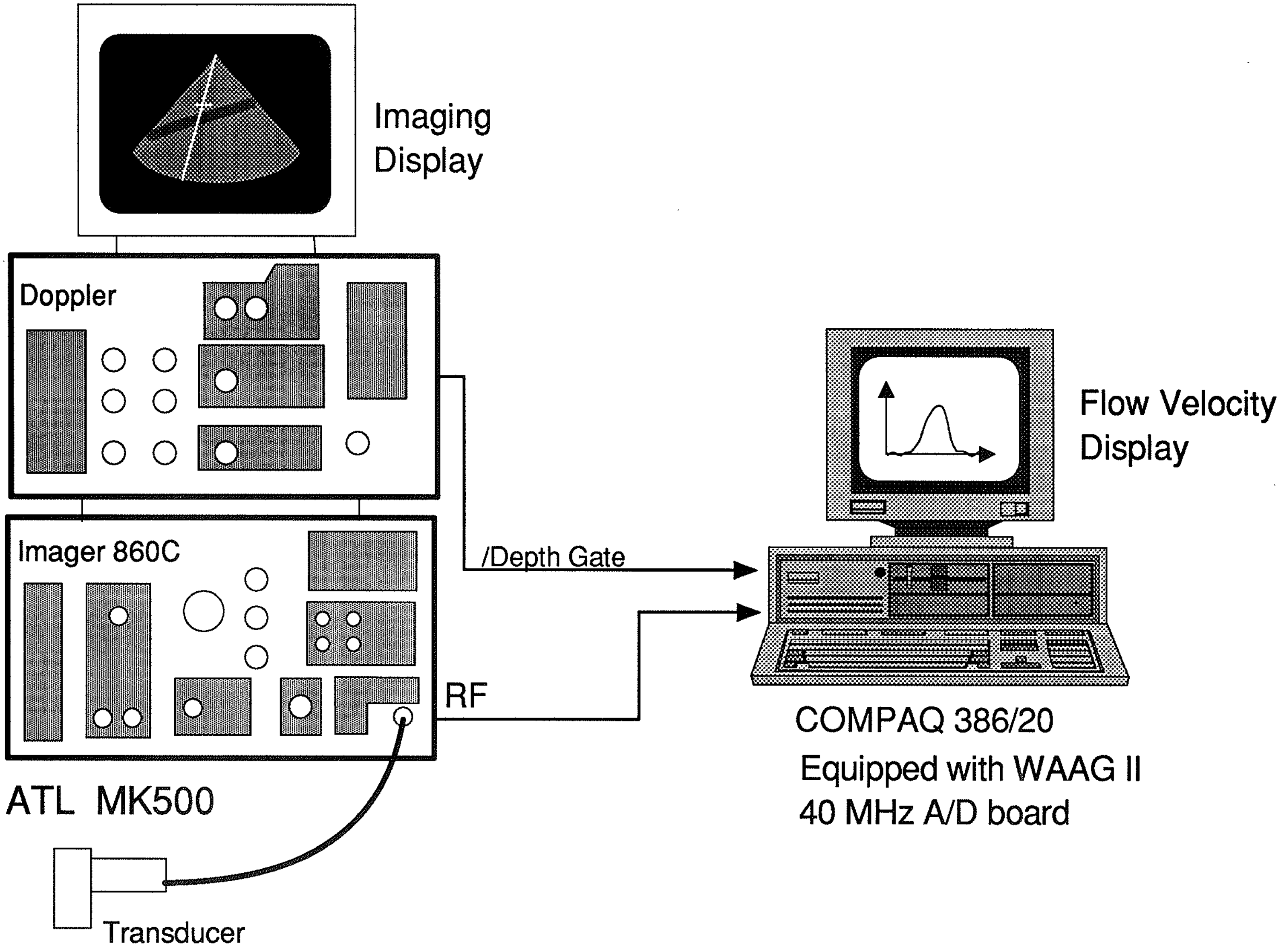


Figure 49 - Temporary UTDC blood flow measurement system. Correlations are done in software and it allows imaging of the blood vessels.

7.1.1 Hardware acquisition of data from the sector display

The MK500 has a set PRF of 1 kHz. The /Depth_Gate signal also occurs at this rate and is used to trigger the WAAG II board to digitize and capture 1K length echoes. At 40 MHz this corresponds to a 19.2 mm length in water (the previous system digitized at a 50 MHz rate, which corresponds to a 15.4 mm length). The 860C mode control knob must be set to the m-line mode and the 5 MHz transducer setting (no rotor elements pressed) must also be selected. The /Depth_Gate signal generated by the pulsed Doppler flow analyzer corresponds to the location of the cursor on the sector display of the ultrasound image, as shown in Figure 50. The cursor can be moved to any point in the image by adjusting the wiper position knob (located on the transducer) and adjusting the sample-volume-depth knob on the pulsed Doppler flow analyzer. Once the cursor is moved to the desired location, the right-foot pedal is pressed which stops the transducer from spinning and freezes its beam along the A-line corresponding to the wiper position (the ultrasound image is also frozen). A key on the COMPAQ's keyboard is pressed which initiates a data aquisition cycle by the COMPAQ and the WAAG II board on the next /Depth_Gate pulse. The next /Depth_Gate pulse will then digitize the RF signal starting at the cursor position for a length determined by /CONVERT. The length of /CONVERT is programmable and for all measurements is set at 1K.

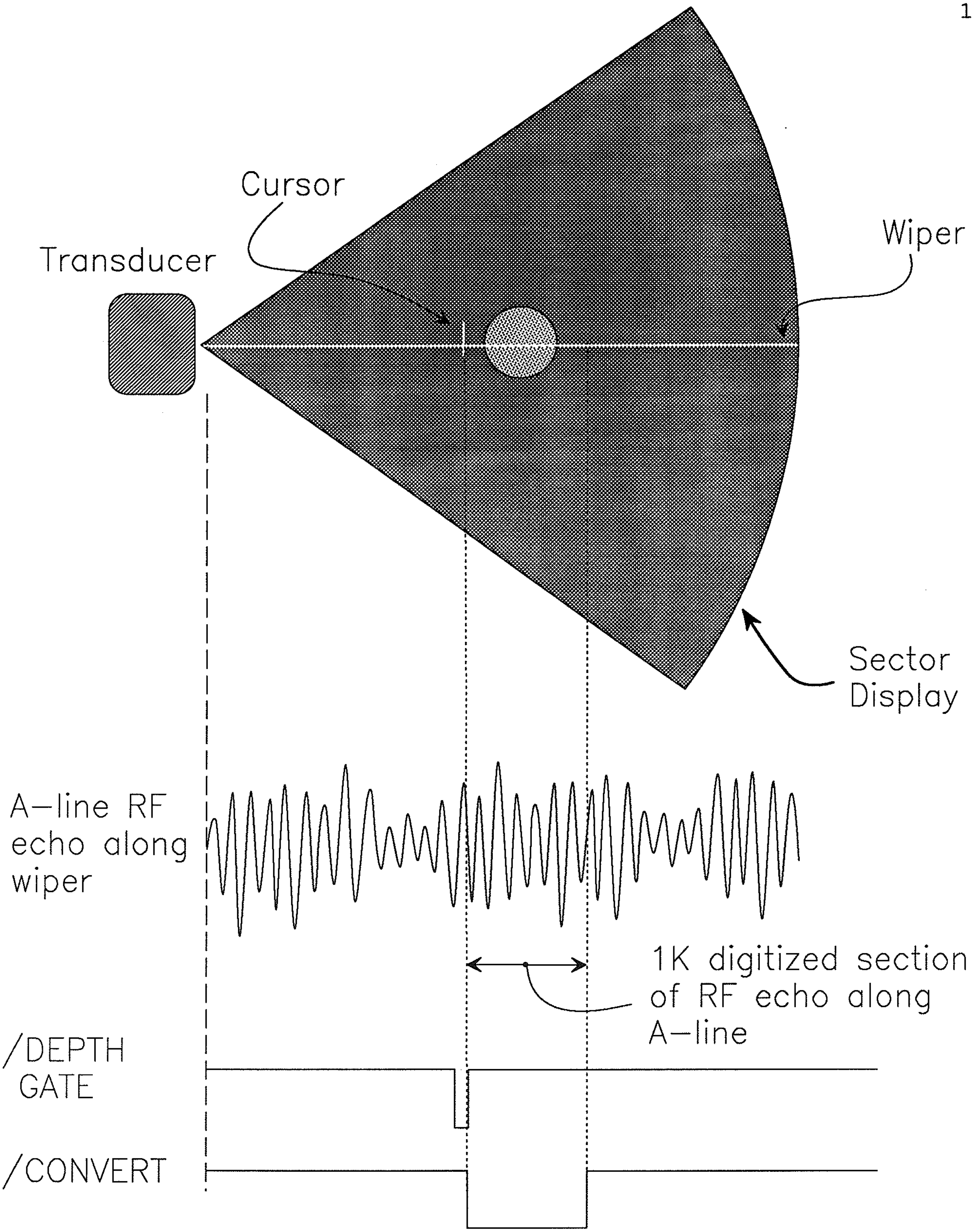

Cursor
Transducer
Wiper
Sector Display
A-line RF echo along wiper
1K digitized section of RF echo along A-line
/DEPTH GATE
/CONVERT

Figure 50 - Relationship between /Depth_ Gate signal and cursor location on the screen. Echoes are digitized at a window 1K wide at the cursor position.

7.1.2 Software

The software written for the temporary system consists of three main programs, one for data aquisition, one for correlations and velocity calculation, and one for plotting. MetaWare's High-C 386 compiler was used for the correlation and velocity calculation since it is extremely quick. Microsoft C version 5.0 was used for the data aquisition program and the graphics program. High-C 386 is approximately seven times faster that Microsoft C for floating-point calculations; unfortunately, at the time of this thesis it had no graphics capability and could not access the WAAG II boards memory because it ran in protected mode. The three programs were linked by a batch file. The data aquisition program captures the RF echoes and writes them to disk. The correlation and velocity calculation program reads in the data from disk, correlates and calculates velocity, and writes the velocity data to disk. The graphics program reads in the velocity data and plots it. When plotting is done, a data and plot file renaming program is executed and then control is transferred back to the data acquisition program. This procedure is awkward, but the process of writing to and from disk is much faster than if the correlation and velocity calculation had been done with Microsoft C. It takes 45 sec from the keypress to the plotting of velocity vs. distance on the screen. The temporary system programs are listed in Appendix E.

7.1.2.1 Data acquisition program

The WAAG II board consists of 128K of circular wrap-around memory [33]. The digitization length is programmed on the board (called the valid trigger count, VTC). Data aquisition consists of arming the board, at which time digitization begins and data are stored in the wrap-around memory. It will keep digitizing, filling the memory and wrapping around until a trigger signal arrives. At the trigger point, the current memory address (called the valid trigger address, VTA) is stored and digitization stops VTC points later. Because of this wrap-around memory structure, it is not possible to acquire consecutive echoes in the WAAG II's memory. Only one echo can be acquired at a time, despite the fact that the board could theoretically store 128 1K echoes. Acquiring echoes thus consists of acquisition of an echo, transferral to PC memory, acquisition of the next echo, transferral to computer, etc. until the desired number of echoes is acquired. The computer memory is then transferred to disk. This process is too slow to acquire echoes at the 1 kHz PRF of the MK500. It takes approximately 1.5 ms to acquire and transfer a 1K echo from the WAAG II board to the PC memory. This means the effective pulse repetition period of the system is 500 Hz, which limits the maximum measured axial velocity to approximately 5 cm/sec. For this system, the number of echoes acquired was twenty.

7.1.2.2 Correlation and velocity calculation program

This program does the correlations and calculates the time shifts for twenty echoes for Δs of 1 to ten. The time shifts for Δ=1 are calculated using echoes 1 and 2, 2 and 3,...,9 and 10; Δ=10 uses echoes 1 and 11, 2 and 12,...,10 and 20; hence, a minimum of twenty total echoes is required. This is a drastic reduction from the 384 echoes used in the previous system. The main reason for choosing twenty echoes is that it reduces processing time considerably, since the correlations are done in software. The results of blood flow phantom experiments indicate that accurate results can be achieved using only twenty echoes.

The time shifts are calculated using the correlation algorithm developed in [9]. Velocities for each Δ are calculated and the weighted average algorithm is used to estimate the final velocity. This program is also capable of calculating the correlations at a higher resolution than for the previous system. The previous system calculated correlations at increments of 0.6 mm. This system can calculate the correlations at increments of 0.75, 0.375, or 0.1875 mm.

7.1.2.3 Graphics program

The graphics program is very simple. It reads in the velocity data and uses the GraphiC 4.1 plotting routines (Scientific Endeavors Corp.) to plot the data.

7.2 Temporary System Experiments

Many constant flow experiments were performed with the blood flow phantom system to validate the temporary system. Since the transducer does not have to be located in a transducer assembly (as with the previous system), the transducer could be oriented in any position with respect to the vessel. Both short-axis and long-axis measurements (as defined in Figure 51) can be made. The 1-D measurements in the long-axis plane were used to validate the system because the transducer angle can be estimated from the long axis image, as shown in Figure 52. A right triangle can be formed by drawing a line from the apex of the sector scan to the vessel which will intersect the vessel at a right angle (line l_p). The measurement angle will be

$$\theta = 90 - \cos^{-1}(l_p/l_w) \tag{47}$$

where l_w is the length of the wiper and l_p is the length of the perpendicular line.

The correlation algorithm is capable of measuring flow velocities in one direction only; hence, the transducer must be oriented such that the component of flow in the direction of the wiper (the measured axial component) is moving away from the transducer, illustrated in Figure 52.

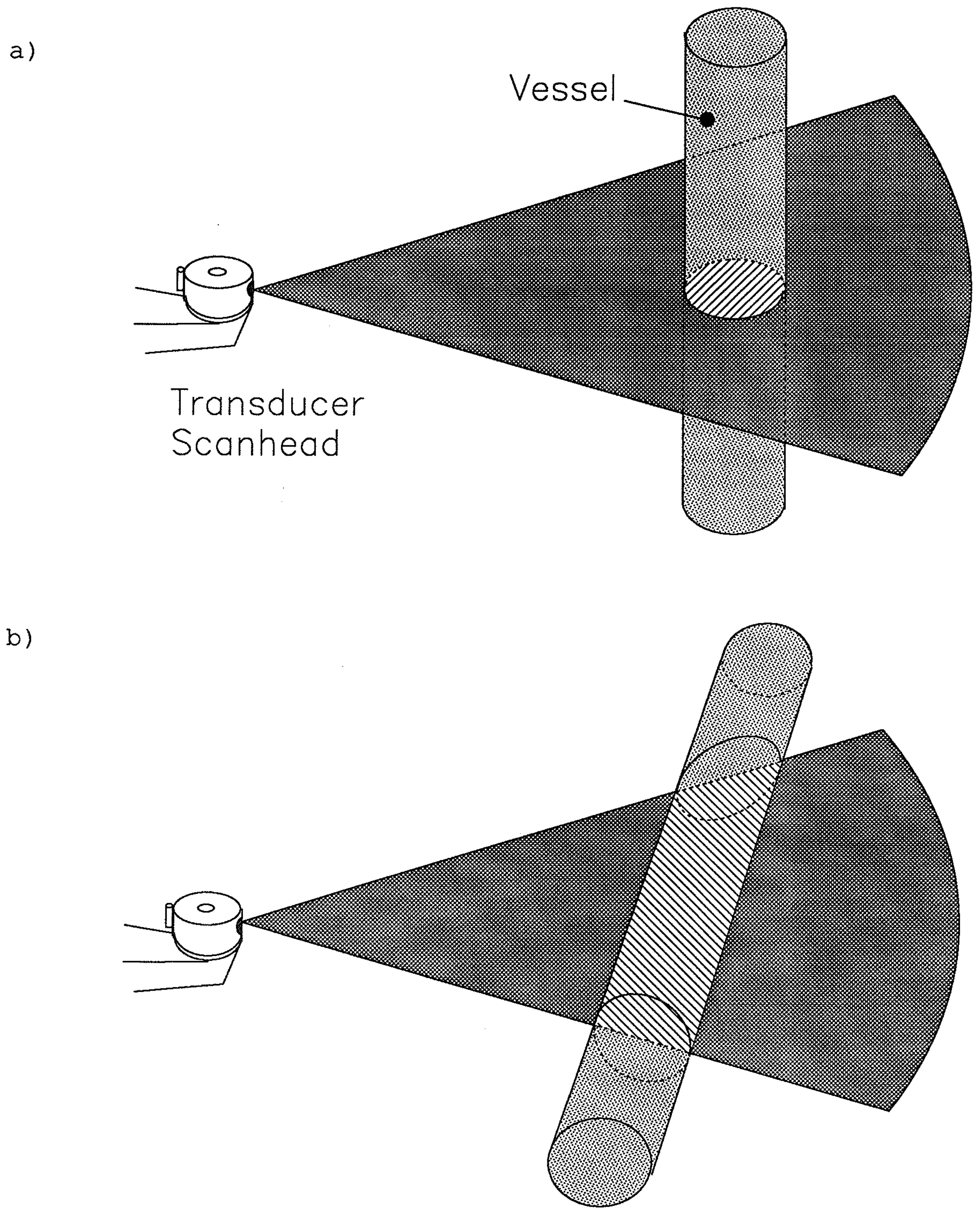


Figure 51 - a) Short axis measurement. The intersection of the sector scan plane and the vessel results in a cross section of the vessel. b) Long axis measurement. The plane of the sector scan is parallel to the vessel plane.

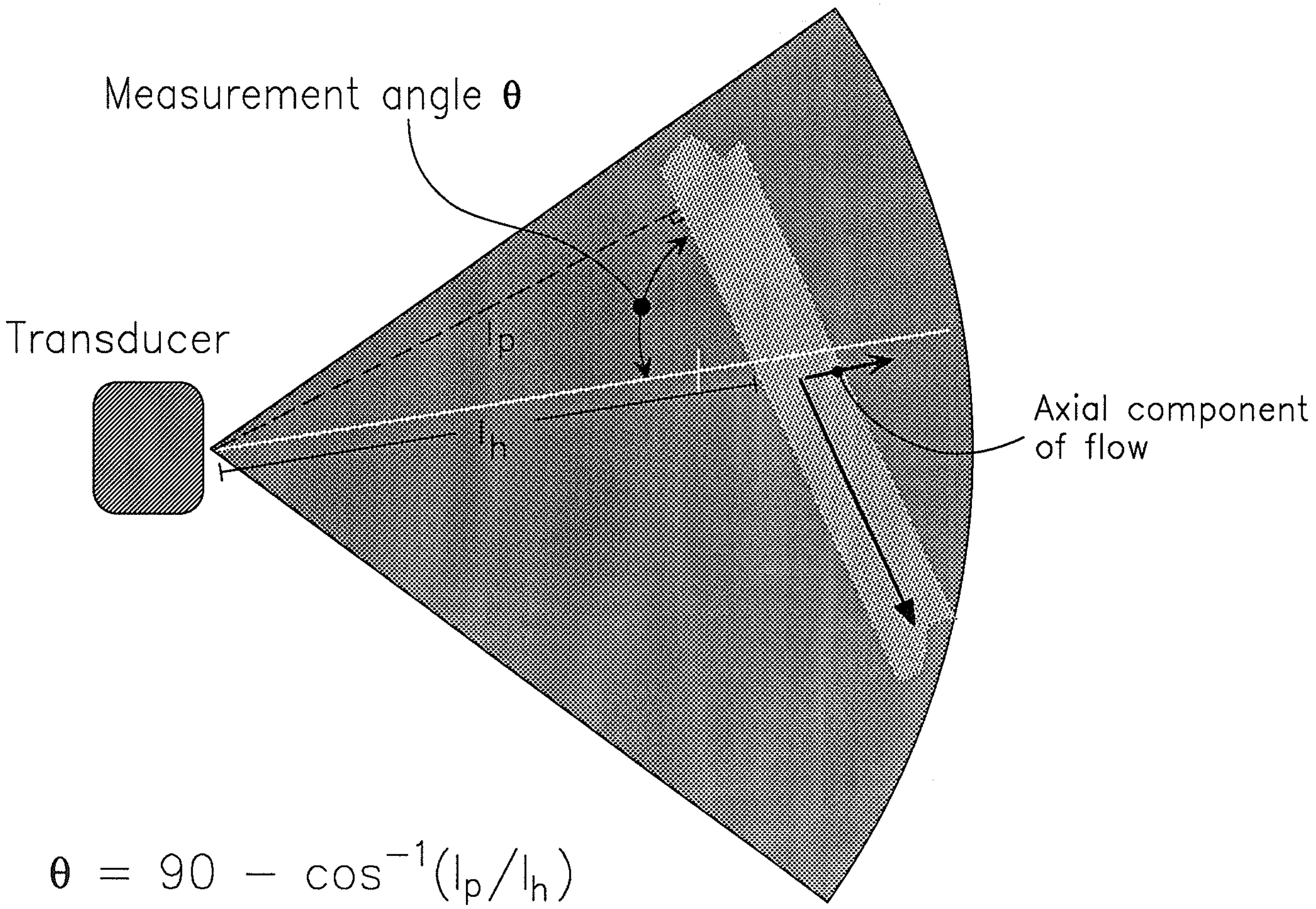

Measurement angle θ
Transducer
l_p
l_h
Axial component of flow
$\theta = 90 - \cos^{-1}(l_p/l_h)$

Figure 52 - The measurement angle θ is the angle between the wiper line and the vessel in a long-axis measurement.

7.2.1 Blood flow phantom measurements

The original transducer assembly was removed from the blood flow phantom since it housed the transducer used by the previous system, which is no longer used. With the previous system, the transducer was immersed totally under water, and the dialysis tubing was perpendicular to the water bath. The ATL MK500 scanhead is not waterproof and cannot be totally immersed in water. In order to image the vessel, the angle at which the dialysis tubing enters the water was changed from 90° to approximately 60°. This allowed the vessel to be imaged by immersing the tip of the scanhead slightly into the water bath, as shown in Figure 53. Measurements were made for constant flow rates in the range of 150 ml/min. Only low flow rates could be measured because of the effective 500 Hz PRF limitation.

Figure 54 shows the long-axis ultrasound image of the vessel. Flow within the vessel is moving from the left to the right, and the wiper is positioned so that the component of flow in the wiper direction is moving away from the transducer. The measured axial velocity can be varied by changing the wiper angle as well as adjusting the hydrodynamic flow rate. The hydrodynamic flow was set at 144 ml/min and the angle was adjusted such that the axial velocities will be under 5 cm/sec. The angle was estimated from the ultrasound image at 77.5°. The temporary system was validated by performing 20 consecutive measurements at the same hydrodynamic flow rate and wiper angle. The volumetric flow was estimated from

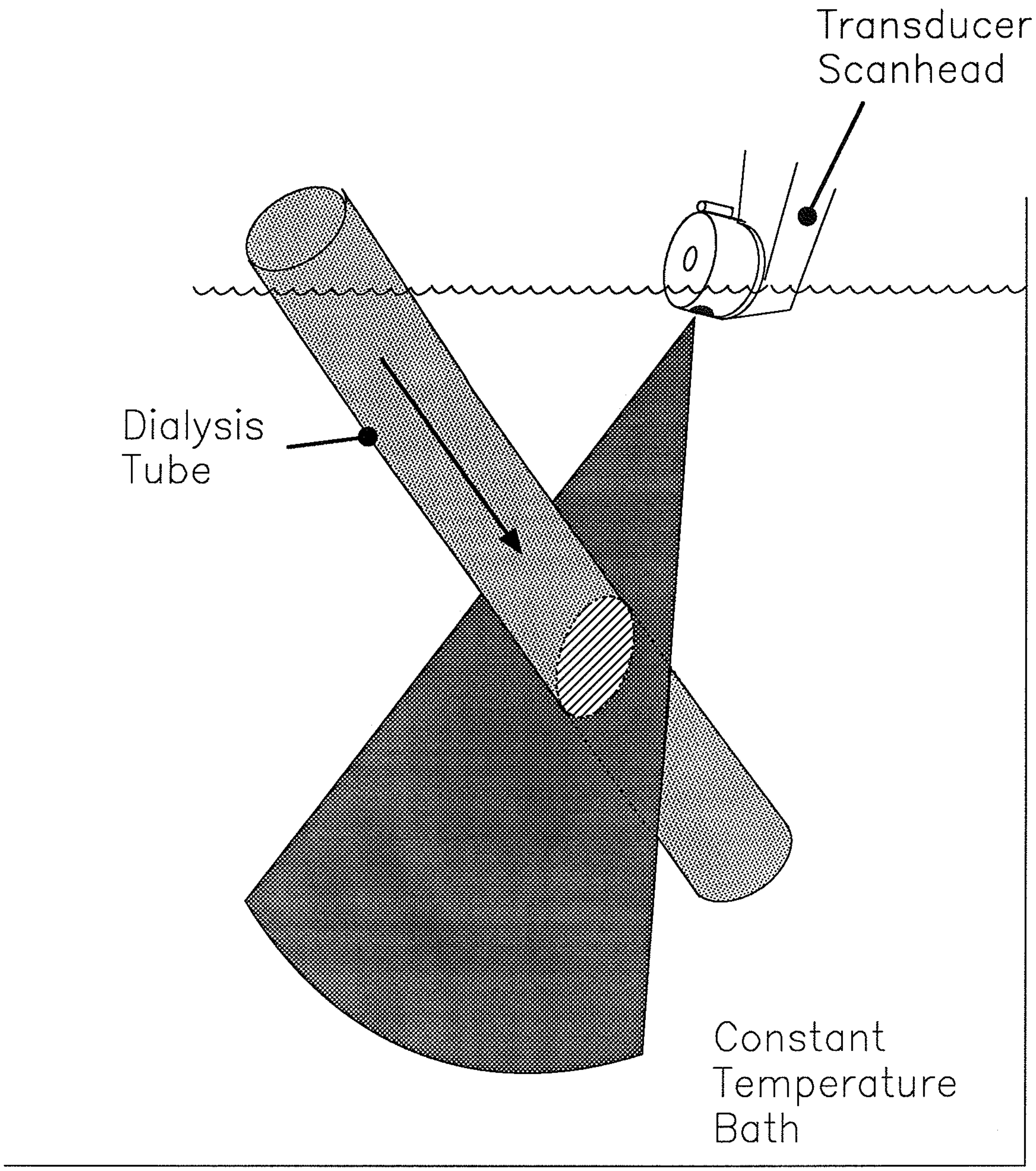


Figure 53 - Imaging of the dialysis tubing in the blood flow phantom with the MK500 scanhead. The tubing was tilted approximately 60 degrees. The scanhead was held solidly in place by a series of clamps.

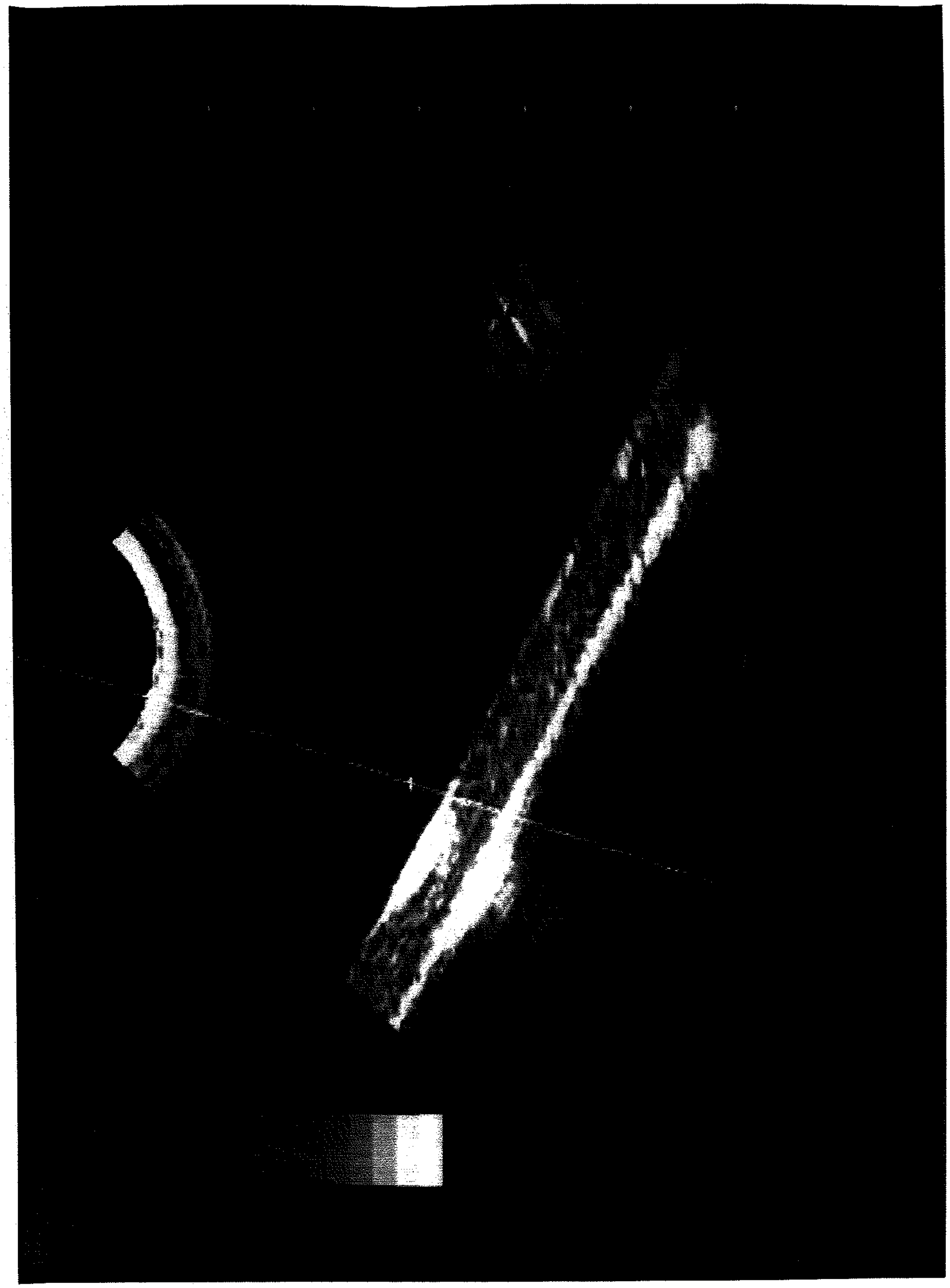


Figure 54 - Long-axis ultrasound image of the 6.5 mm dialysis tubing. The hydrodynamic flow rate was 144 ml/min and the measurement angle calculated from this image was 77.5 degrees.

the 1-D velocity profiles as described in Chapter 6, Section 2. Table 3 lists the 20 measurements, the estimated volumetric flow, and the error referenced to the hydrodynamic flow.

Table 3: Constant flow experiments with temporary system. The hydrodynamic flow rate = 144 ml/min.

Measurement #	Calculated flow ml/min	Error %
1	142.5	- 1.04
2	156.3	+ 8.50
3	128.7	-10.60
4	138.0	- 4.16
5	147.1	+ 2.12
6	165.5	+14.90
7	156.3	+ 8.54
8	156.3	+ 8.54
9	151.7	+ 5.34
10	137.9	- 4.23
11	151.8	+ 5.14
12	142.6	- 0.90
13	142.6	- 0.90
14	156.3	+ 8.50
15	174.8	+21.30
16	131.0	- 9.0
17	161.0	+11.8
18	144.9	+ 6.3
19	147.2	+ 2.2
20	175.0	+21.5

average error = +4.4%
standard deviation = 8.68%

In general, the measurement error for these experiments was within 15%, except for measurements 15 and 20, which had a 21% error. Fifteen out of the 20 measurements had an accuracy better than 10%. The reported accuracy of the previous system for constant flow was 15%; these measurements are well within this limit.

The velocity vs. range was plotted for each measurement in Table 3 for low (0.75 mm) and high (0.1875 mm) resolutions. Most plots displayed a parabolic shape for both resolutions. Figure 55 shows velocity vs. range plots for a measurement which shows a nice parabolic shape at low resolution (Figure 55a)) but doesn't look so smooth at high resolution (Figure 55c)). The same RF echo data were used to generate the three plots in Figure 55. These plots indicate that a higher resolution may be needed to get a true plot of flow within the vessel. For very small vessels, the higher resolution will become necessary since only one or two points would be sampled at low resolution. An experiment using a 1.5 mm diameter vessel was performed with the real-time system described in Chapter 8, Section 2.5, and is discussed there. The disadvantage of using the higher resolution is that it takes proportionally longer to get the result. The high-resolution plot takes 3 min to generate.

7.2.2 *In vivo* velocity measurements

A number of *in vivo* velocity measurements were performed on relatively large canine subjects. Collaboration with a veterinarian was essential to procure the subjects and anesthetize them during the measurements. The femoral vein and artery were chosen as the vessels to be measured due to their size and close proximity to the skin. Since it can be very difficult for someone

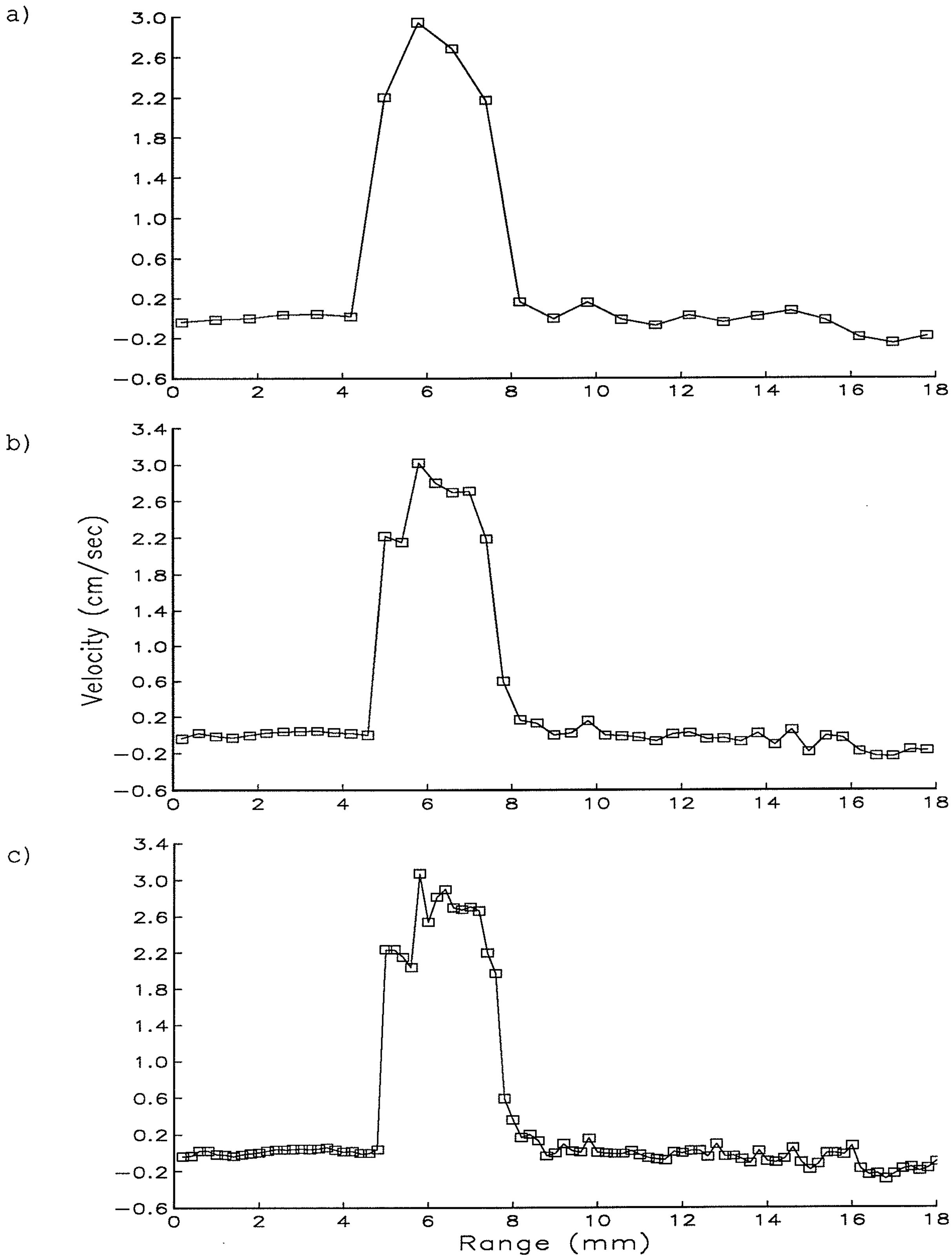


Figure 55 - Velocity vs. range plotted at different resolutions for the same data. a) 0.75 mm (default), b) 0.375 mm and c) 0.1875 mm.

inexperienced in looking at ultrasound images to locate the vessels of interest, the assistance of two sonographers was enlisted to help locate the vessels from the ultrasound image. The Doppler flow analyzer was also used to confirm the location of a vessel by listening to the Doppler audio output. The measurement procedure consisted of locating a vessel and moving the cursor to the middle of the vessel. The 860C imager was then switched to duplex mode to listen to the Doppler output. A vein produces an audio Doppler output which sounds like a constant whooshing noise; an artery produces a pulsating whoosh as the flow varies with the cardiac cycle. Once the location of a vessel had been confirmed, the 860C imager was put back into m-line mode and the cursor was moved out of the vessel to a point slightly before the front vessel wall. The foot pedal was depressed, freezing the transducer onto the wiper A-line and the keyboard was pressed. Forty-five sec later the velocity vs. distance plot appeared on the computer screen.

7.2.2.1 Canine femoral vein measurement

Figure 56 shows the results of a femoral vein measurement. The short-axis ultrasound image is shown in Figure 56a) and the un-angle corrected velocity vs. range plot is shown in Figure 56b). The curving white line in Figure 56a) is due to the skin of the animal, and the large dark area between the skin and the transducer is due to the acoustical standoff. The white hash

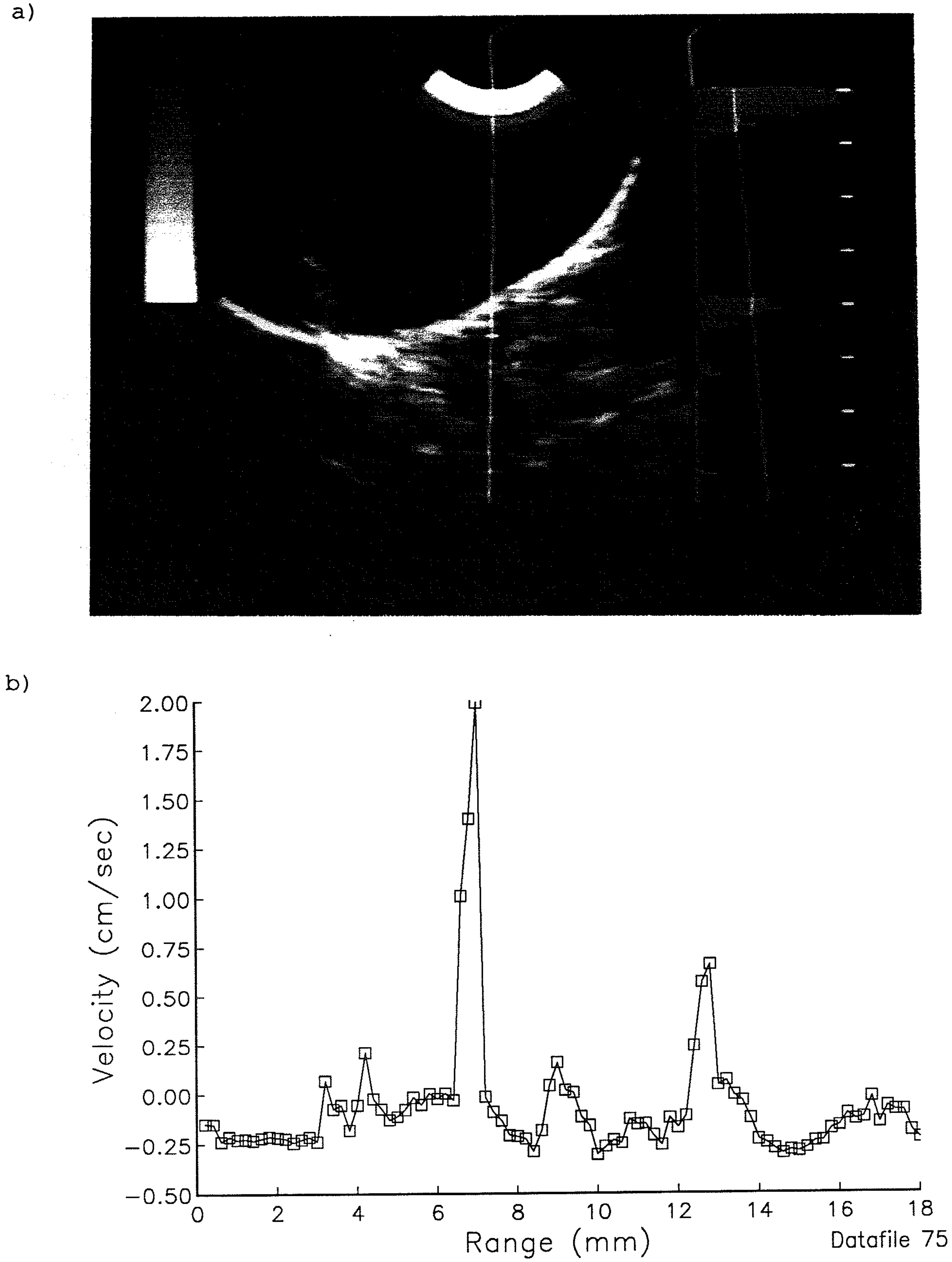


Figure 56 - a) Short-axis ultrasound image of a canine femoral vein.
b) Velocity profile of the vein.

marks at the far right of the image represent one centimeter increments in distance. It is fairly difficult for an inexperienced viewer to locate the vein. The position of the vein is within the dark area just past the cursor. The vein collapses rather easily due to pressure from the transducer and acoustical standoff; hence, the distance from the front wall to the rear is very small. This is evident from looking at the velocity vs. range plot in Figure 56b). It shows a velocity peak approximately 6 mm from the cursor, and a width of about a millimeter. This velocity profile and vessel width are normal for the femoral vein of an anesthetized dog [34].

7.2.2.2 Canine femoral artery measurement

Figure 57 shows the results of a long-axis measurement on the femoral artery (at an unknown part of the cardiac cycle). The artery is easier to locate since it retains its shape (the flow pressure is much greater on the arterial side than the venous) and is the thick black horizontal line just below the cursor. Flow in the artery is on average moving from right to left. The velocity vs. range plot (not angle corrected) shows the width of the vessel to be approximately 2.25 mm, and also shows multiple peaks. This may be due to turbulent flow within the artery. The maximum velocity is 2.13 cm/sec. The angle between the vessel and wiper is estimated at 47.8°, which makes the angle-corrected maximum velocity 3.17 cm/sec. The location of the artery with

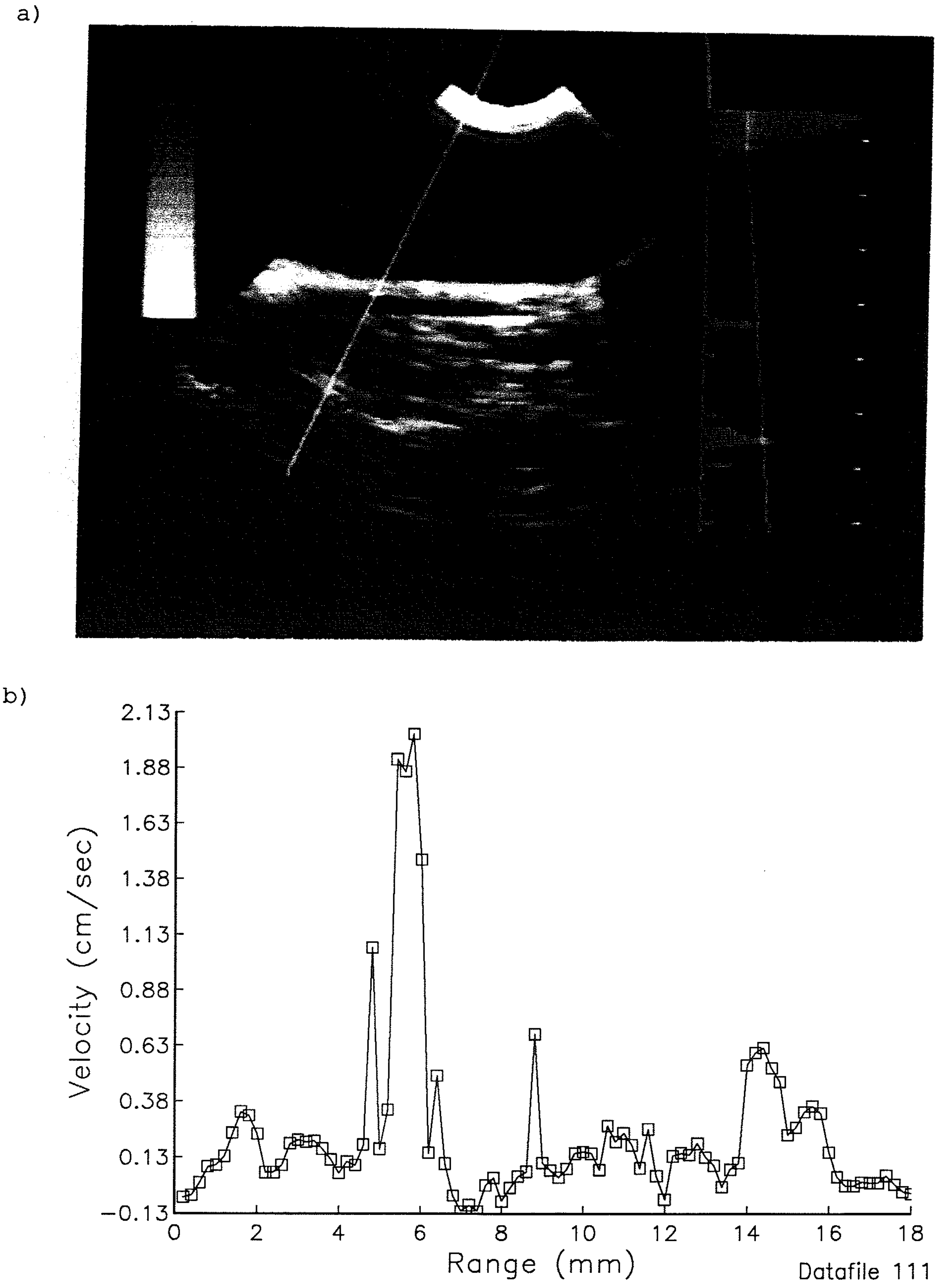


Figure 57 - a) Long-axis ultrasound image of a canine femoral artery. b) Velocity profile of the artery.

respect to the cursor does not quite agree. The cursor appears to be positioned directly at the front wall of the artery, while the velocity vs. range plot shows the location of the artery approximately 4.5 mm from the cursor. The ultrasound image is frozen when the foot pedal is pressed. If there is movement between depression of the foot pedal and striking of the key, it would not show up on the image. The transducer was most likely moved slightly between depression of the foot pedal and striking of the keyboard (it is being held by a human, not held in place by a solid fixture). The velocity magnitude, vessel diameter, and flow profile shown in Figure 57b) are within the range of those found in normal anesthetized dogs [34].

7.3 Summary

This temporary system proved the usefulness of having an ultrasound imager available to locate the vessel. The presence of the Doppler audio output helped confirm the location of a vessel. A major disadvantage of the system is that it is not real time, and the process of locating the vessel, hitting a key, and waiting 45 seconds for the output is very cumbersome. This 45-second delay would be unacceptable in a clinical setting. There is the possibility that the transducer may be moved between the pressing of the foot switch and depressing of the key (as with the femoral artery measurement), and it would be very difficult to hold the transducer in the same place for multiples of 45 seconds to make repeated measurements at a given vessel. The real-time system described in the next chapter alleviates these problems.

CHAPTER 8

REAL-TIME SYSTEM

This chapter describes the real-time system. The correlations are done in hardware via a custom-designed residue-number system (RNS) correlator. It is capable of producing a velocity vs. range plot in 0.7 seconds. The velocity plots are produced continuously (no key has to be pressed as with the temporary system). This system has been validated using the blood flow phantom under the same conditions used for the previous system and the temporary system, as well as with attenuating media, multiple vessels, and with smaller vessel sizes. *In vivo* measurements in the human carotid artery have also been made.

8.1 Real-Time System Setup

Figure 58 shows the setup of the real-time system, which consists of the ATL MK500 imager, a COMPAQ computer, and a custom-built ultrasound data acquisition and residue number correlator (UDA-RNC) system. The UDA-RNC system consists of a TRW 50 MHz A/D and bus expander interfaced to the RNS correlator. The same RF and /Depth_Gate signals used by the temporary system are used by the real-time system. In addition, the cursor position and wiper

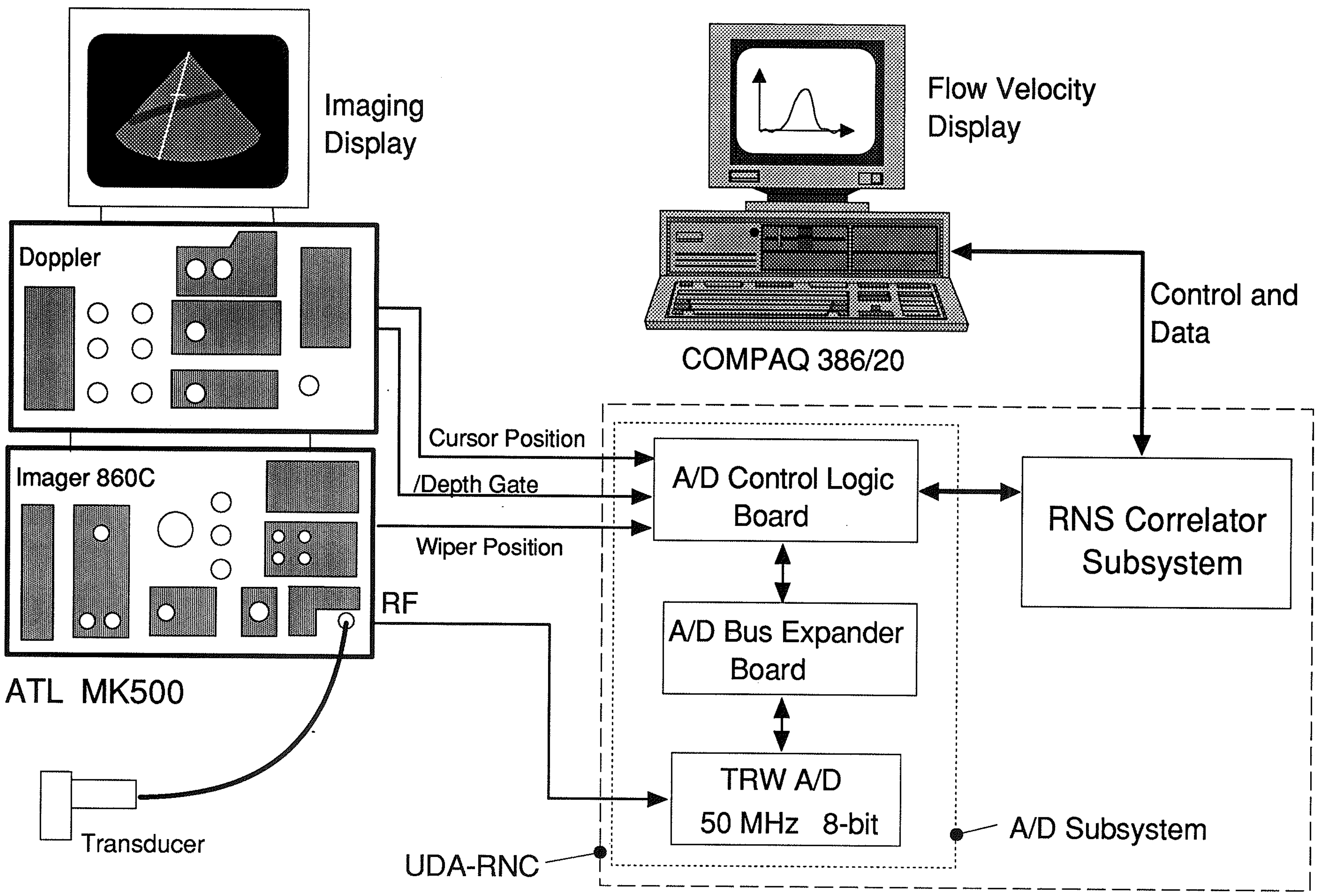


Figure 58 - Real-time ultrasound time-domain correlation blood flow measurement system.

position signals have also been tapped from the MK500. The RF signal is digitized and placed into the RNS correlator memory. The correlator performs the correlations and produces time shift values, which are sent to the COMPAQ. The COMPAQ performs velocity calculations from the time shifts and produces the velocity vs. range plots.

8.1.1 Acquisition of data from the sector display

The acquisition of data from the sector display is essentially the same as that of the temporary system (as illustrated in Figure 50). The /Depth_Gate signal from the MK500 triggers the A/D to capture 1K echoes. At 50 MHz this corresponds to a 15.4 mm length in water. The 5 MHz transducer setting must be chosen (no rotor element buttons depressed on the 860C). The 860C must be set to m-line mode and the cursor moved to the desired location on the sector display. Once there, the foot pedal is pressed which freezes the beam in the direction of the wiper. The velocity vs. range is continually updated on the COMPAQ's monitor. The velocity vs. range plots on the COMPAQ's screen are only valid when the foot pedal is depressed.

8.1.2 Axial velocity measurement limits

The maximum measurable axial velocities are determined by the unchangeable PRF of the ATL MK500, which is 1 kHz. This limits the measurable axial velocity to less than 10 cm/sec. In order to measure flow velocities greater than 10 cm/sec, the transducer measurement angle must be adjusted such that the axial component of the flow velocity vector is less than 10 cm/sec. If the flow is very fast, a very high measurement angle, sometimes larger than 80°, needs to be selected. The axial velocity component is directly proportional to the cosine of the measurement angle θ. The value of $\cos(\theta)$ changes rapidly at high angles, especially in the range of 80° - 90°. This makes the adjustment of the wiper angle somewhat tricky, especially for *in vivo* measurements where the transducer is not held rigidly in place as it is in the blood flow phantom.

8.1.3 UDA-RNC system

The ultrasound data acquisition and residue number correlator systems were a joint project. The ultrasound data acquisition system was constructed by the author of this thesis and the residue number system correlator was designed and constructed by Jerome Chen [36]. The UDA-RNC system is constructed in an S100 card cage and the A/D and correlator communicate via the S100 bus.

8.1.3.1 A/D subsystem

The A/D subsystem is based on the bus-expander design used in [31]. It contains registers programmable from the COMPAQ which control the echo length to be digitized (CONVERT in Figure 50), the number of echoes to digitize, and the location in correlator memory to put the echoes. The A/D subsystem is capable of acquiring and storing consecutive echoes, unlike the WAAG II board in the temporary system. The RNS correlator subsystem contains 256K of dual port memory; thus, the A/D is capable of acquiring 256 consecutive 1K echoes. The echoes are acquired at the 1 kHz PRF rate of the MK500 system. The schematics and timing diagrams for the A/D subsystem are in Appendix F.

8.1.3.2 RNS correlator subsystem

The RNS correlator subsystem was designed and constructed by Jerome Chen and described in detail in [36]. The RNS correlator performs the following basic 40-point digital correlation on two echoes:

$$R(a,b) = \sum_{i=0}^{39} E_1(a+i) * E_2(b+i) \tag{48}$$

where R(a,b) is the correlation value between two echoes where E_1 is the first echo, a is the location of the correlation window within the first echo, E_2 is the second echo, and b is the location of the correlation window within the second echo.

8.1.3.3 Background of residue number system arithmetic

RNS arithmetic is extremely efficient in performing high-speed addition and multiplication and is ideally suited for the

UTDC digital correlation function [37]. The concept behind using RNS arithmetic is to replace hardware multiplications (which are time-consuming) by addition and high-speed lookup tables stored in ROM. RNS accomplishes this by breaking down data words of a given length to a number of smaller length words called residues. The number and size of the residues are dependent on the desired dynamic range of the system. These residues are integers of different moduli and must be pairwise prime. A data word is encoded into residues of these moduli by the equation

$$x_i = \begin{cases} |X|_{\mathrm{mod}\ p_i} & x \geq 0 \\ p_i - |X|_{\mathrm{mod}\ p_i} & x < 0 \end{cases} \qquad (49)$$

where X is the original length data word, the p_i's are the moduli, and the x_i's are the residues of the moduli. Residues can be reconstructed back into a natural number using the Chinese Remainder Theorem:

$$Y = \left| \sum_{i=1}^{L} m_i \left| m_i^{-1} y_i \right|_{\mathrm{mod}\ p_i} \right|_{\mathrm{mod}\ M} \qquad (50)$$

where Y is the natural number base 10, L is the total number of moduli, $M = p_1p_2p_3$, $m_i = M/p_i$, and m_i^{-1} is the inverse of m_i such that $|m_i m_i^{-1}|_{\mathrm{mod}\ pi} = 1$.

For this application, the data word length is 8 bits and the correlation window is 40 samples wide. A dynamic range of 21 bits (8-bit signed value * 8-bit signed value + 40 multiplied values) is required. This can be covered by four 6-bit moduli: 64, 63, 61, and 59. These moduli were chosen so that high-speed 8192 x 8 jROMS with a 13-bit address space can be used for the lookup tables [36].

As an example of RNS correlation, suppose that the following simple two-sample correlation with the above moduli is to be performed:

$$(109)*(47) + (-83)*(12) = 4127 \qquad (51)$$

The moduli and parameters are

$$\begin{array}{lll} p_1 = 59 & m_1 = 245{,}952 & m_1^{-1} = 31 \\ p_2 = 61 & m_2 = 237{,}888 & m_2^{-1} = 5 \\ p_3 = 63 & m_3 = 230{,}336 & m_3^{-1} = 8 \\ p_4 = 64 & m_4 = 226{,}737 & m_4^{-1} = 17 \end{array}$$

$$M = 14{,}511{,}168$$

The residues for all 6-bit numbers are precalculated (from Equation (49)) and stored in ROMs, as shown in Table 4 (shown only for values used in Equation (51)):

Table 4: Six-bit ROM lookup table.

Value	MOD 59 ROM	MOD 61 ROM	MOD 63 ROM	MOD 64 ROM
12	12	12	12	12
47	47	47	47	47
109	50	48	46	45
-83 (173)	35	39	43	45

Thus the natural numbers in terms of their moduli are

$$\begin{aligned} 109 &= \{\ 50_{59},\ 48_{61},\ 46_{63},\ 45_{64}\ \} \\ 47 &= \{\ 47_{59},\ 47_{61},\ 47_{63},\ 47_{64}\ \} \\ -83 &= \{\ 35_{59},\ 48_{61},\ 43_{63},\ 45_{64}\ \} \\ 12 &= \{\ 12_{59},\ 12_{61},\ 12_{63},\ 12_{64}\ \} \end{aligned}$$

where each residue number within brackets is a 6-bit value. The correlation in the RNS domain becomes

$$\begin{aligned} y_{59} &= (\ 50*47 + 35*12\ )_{59} \\ y_{61} &= (\ 48*47 + 39*12\ )_{61} \\ y_{63} &= (\ 46*47 + 43*12\ )_{63} \\ y_{64} &= (\ 45*47 + 45*12\ )_{64} \end{aligned} \tag{52}$$

All of the multiplies in Equations (52) are done using lookup tables stored in ROM, as illustrated in Figure 59. The two 6-bit numbers to be multiplied form an effective 12-bit address of a 4096 x 6 bit ROM. The value stored at the 12-bit address is the result of the multiplication of the individual two 6-bit residues. After the lookup-table multiplications, Equations (52) become

$$\begin{aligned} y_{59} &= (\ 49 + 7\ )_{59} = 56_{59} \\ y_{61} &= (\ 60 + 41\ )_{61} = 40_{61} \\ y_{63} &= (\ 20 + 12\ )_{63} = 32_{63} \\ y_{64} &= (\ 3 + 28\ )_{64} = 31_{64} \end{aligned} \tag{53}$$

where y_{nn} are the residues of the result of the correlation. The natural result Y can then be computed using from Equation 50:

$$\begin{aligned} Y &= \left| 245{,}952*(31*56)_{59} + 237{,}888*(5*40)_{61} + 230{,}336*(8*32)_{63} + 226{,}737*(17*31)_{64} \right|_{\text{mod } 14{,}511{,}168} \\ &= \left| 6{,}148{,}800 + 4{,}044{,}096 + 921{,}344 + 3{,}401{,}055 \right|_{\text{mod } 14{,}511{,}168} \\ &= \left| 14{,}515{,}295 \right|_{\text{mod } 14{,}511{,}168} \\ &= 4127\ \checkmark \end{aligned}$$

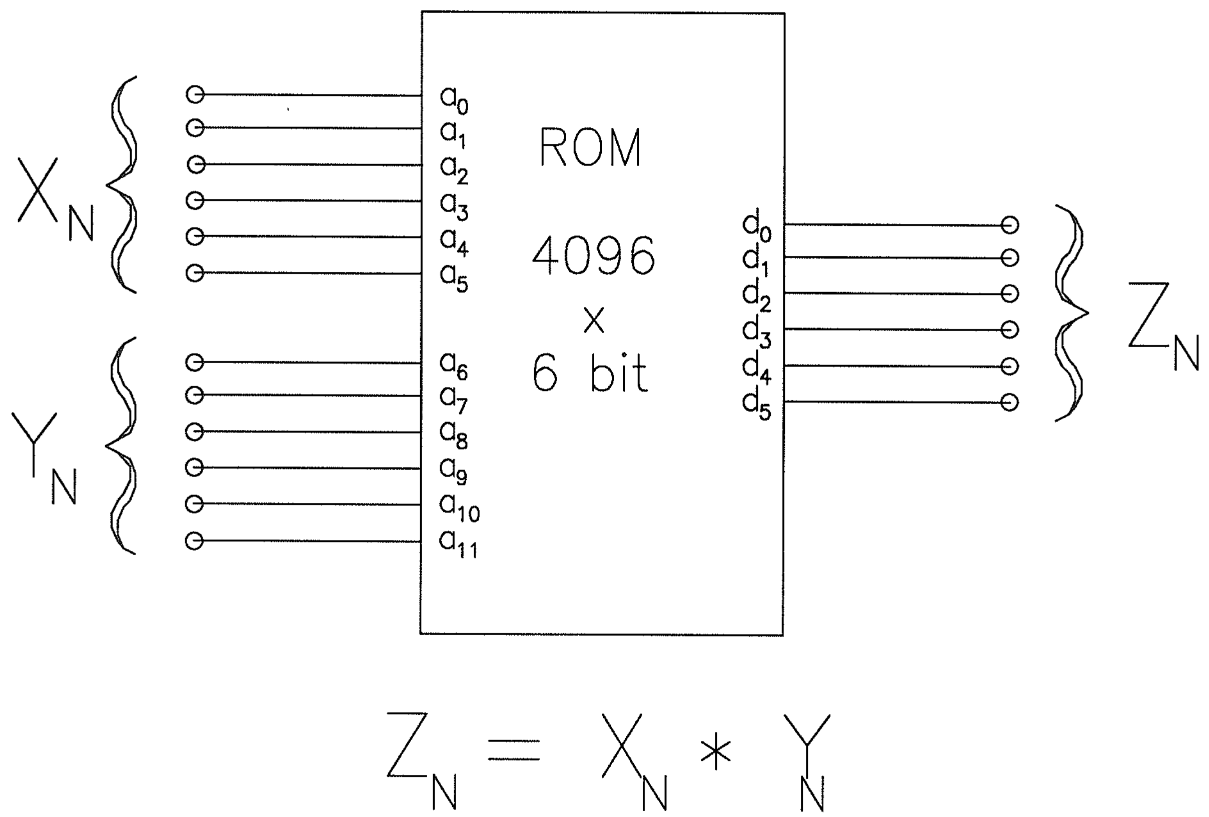


Figure 59 - A 4096 x 6 bit ROM can be used to perform the modulus multiplication of two 6-bit numbers of modulus N. The result Z is precalculated and stored in the ROM for all combinations of X and Y.

Thus all of the multiplications required in the correlation process have been replaced by additions and lookup tables.

8.1.4 Software

A tremendous amount of flexibility has been incorporated into the software controlling the real-time UTDC flowmeter system. The number of echoes, the echo spacing Δ, and the effective pulse repetition period can all be varied to examine the effects of each on the accuracy and the speed of the velocity vs. range estimate. Once chosen, the system is essentially free-running; that is, the velocity vs. range is continuously updated to the screen without any further operator intervention. The velocity data as well as the actual RF echo data can be captured and stored to disk. The RF data can also be loaded into the correlator memory from disk. This allows the velocity to be calculated using different parameters (such as Δ_{max} and number of echoes) for the same RF data to examine the effects of the parameters.

8.1.4.1 UTDC flowmeter program

The entire UTDC flowmeter program has been written using Borland Turbo-C. This was found to run approximately two times faster and compile three times faster than Microsoft C 5.0. As with Microsoft C, Turbo C also runs the 386 microprocessor in 286 emulation and is therefore not as fast as Metaware High-C 386. It has graphics capability and can access hardware ports which High-C 386 was not capable of doing. (Since the time of this thesis, Metaware has released a newer version of High-C 386 capable of graphics and accessing hardware ports.) Since the bulk of the correlation process is done in hardware, the difference in running time of the flowmeter program under Turbo-C instead of High-C 386 is minimal. Future versions of the UTDC flowmeter should, however, be ported over to the latest version of High-C 386 for optimum speed. The UTDC flowmeter program is listed in Appendix G.

8.1.4.2 Correlation algorithm

The hardware RNS correlator performs the windowed correlation as expressed in Equation (48). The normalized correlation coefficient, Equation (36), is calculated in software after three calls to the RNS correlator: one for the numerator and two for the denominator of Equation (36). The two sum of squares terms in the denominator are basically echoes correlated with themselves. The maximum in the normalized correlation coefficient R(s) is determined over blocks of 20 echoes using the algorithm developed by Embree [9] (see function corr_20 in Appendix G). Δ_{max} can be varied from 1 to 10, and 20 echoes is the minimum number of echoes needed for $\Delta_{max} = 10$ (see Chapter 7, Section 1.2.2). For $\Delta_{max} = 1$, the correlations are performed only for adjacent echoes and no weighted averaging takes place. For a 20 echo block, $\Delta_{max} = 1$ averages over 18 echo pairs; Δ_{max}=10 averages over 90 echo pairs.

The number of 20-echo blocks can also be chosen from software. The velocity results from the individual 20-echo blocks are then averaged to form the final velocity result.

8.1.5 Speed of real-time system

The speed of the real-time system depends on the particular processing parameters used. The system runs quickest with $\Delta_{max}=1$ since no weighted averaging scheme is used. For 210 echoes and $\Delta_{max}=1$, the real-time system updates the velocity vs. range plot every 0.7 sec. The 210 echoes produces 209 individual velocity values which are equally averaged. A weighted averaging estimate using a block of 20 echoes and $\Delta_{max}=10$ produces 90 echo pairs and requires approximately 1.6 sec to update the velocity display. A $\Delta_{max}=10$ and 210 echoes produce 1800 echo pairs and required 35 sec to produce a velocity profile. The number of echoes can be varied from 20 to 250, and Δ_{max} can be varied from 1 to 10, with the processing time varying accordingly with the parameters chosen. The most useful parameter settings were found to be $\Delta_{max}=1$ with 210 echoes and $\Delta_{max}=10$ with 20 echoes for real-time applications and $\Delta_{max}=10$ with 210 echoes for slower post-processing applications. Section 8.2.4 examines the effects of these parameter settings on the accuracy of the volumetric flow estimate.

8.2 Blood Flow Phantom Experiments

The real-time system has been tested under a number of different conditions with the blood flow phantom system. It has been verified with the 6.5 mm dialysis tubing at different flow rates, transducer angles, and power settings of the MK500. Experiments have also been performed with an attenuating/reflective material between the transducer and the vessel, with smaller size vessels, and also with two vessels (one behind the other with respect to the transducer).

8.2.1 Validation with 6.5 mm dialysis tubing

The real-time system was validated in the blood flow phantom system in the same manner that the temporary system was. The transducer was mounted in a fixed position in the long axis measurement mode. The hydrodynamic flow was varied from approximately 75 ml/min to 300 ml/min, and the transducer angle was varied between 55° and 85°. Twenty consecutive measurements were recorded and the volumetric flow calculated as described in Chapter 6, Section 2. These validations were performed in real time with processing parameters Δ_{max}=1 and echoes=210. Table 5 lists the results for two measurement sets made at 74.5 ml/min $\angle 56.9^{\circ}$ and 300.0 ml/min $\angle 85.0^{\circ}$. The errors at 74.5 ml/min were within 17% (except for measurement 10 with an error of +23.1%). The average error over the 20 measurements was calculated to be -2.7% with a standard deviation of 11.0%. The errors at 300

Table 5: Real-time flow measurements in 6.5 mm dialysis tubing.

	Flow = 74.5 ml/min
	Angle = 56.9 deg
			Flow = 300.0 ml/min
			Angle = 85.0 deg	

Measurement Number	Measured Flow ml/min	Error %	Measured Flow ml/min	Error %
1	75.0	+ 0.7	294	- 1.9
2	86.8	+16.6	314	+ 4.6
3	72.5	- 2.6	302	+ 0.5
4	62.8	-15.6	307	+ 2.3
5	62.1	-16.6	311	+ 3.7
6	64.1	-14.0	238	-20.7
7	78.9	+ 6.0	212	-29.4
8	74.6	+ 0.1	302	+ 0.7
9	81.2	+ 9.1	282	- 5.9
10	91.7	+23.1	279	- 6.8
11	74.3	- 0.2	271	- 9.7
12	73.0	- 2.0	305	+ 1.8
13	76.8	+ 3.1	208	-30.6
14	65.3	-15.0	313	+ 4.1
15	66.7	-10.5	374	+24.7
16	67.9	- 8.8	235	-21.7
17	69.3	- 7.0	310	+ 3.4
18	69.2	- 7.1	307	+ 2.5
19	61.8	-17.0	206	-31.2
20	76.6	+ 2.9	258	-14.0
	average error =	- 2.7%	average error =	-6.2%
	std deviation =	11.0%	std deviation =	14.5%

ml/min and 85° were somewhat worse. The average error was -6.2% with a standard deviation of 14.5%. Six of the measurements had errors greater than 20%, with the maximum at -31.2%. These greater errors are most likely due to the large measurement angle of 85°. A high measurement angle was required to get the axial components of the flow velocity in the range measurable by the ATL MK500's 1 kHz PRF rate. Previous research has shown that the precision of the time-domain correlation technique is dependent on the transducer measurement angle, with the precision becoming progressively worse at high angles. These results are directly in accordance with that research. In general, the accuracy of the real-time system is within the same range as previous systems.

8.2.2 Measurements at different power levels

The ATL MK500 860C imager is equipped with an energy output knob which is calibrated in decibels. It has a minimum setting of 0 dB and a maximum of 16.5 dB and can be adjusted in increments of 1.5 dB. To examine the effects of different power levels, the hydrodynamic flow rate was set at 111.3 ml/min with a corresponding measurement angle θ=74.6°. The ultrasound image (5 MHz) is shown in Figure 60 for the maximum power output and minimum power output, and the digitized RF signals (1K in length) reflected from the vessel are shown in Figures 61 and 62 for the five available power settings of 16.5, 12.0, 7.5, 3.0, and 0.0 dB. The locations of the vessel walls in the RF signals of Figures 61

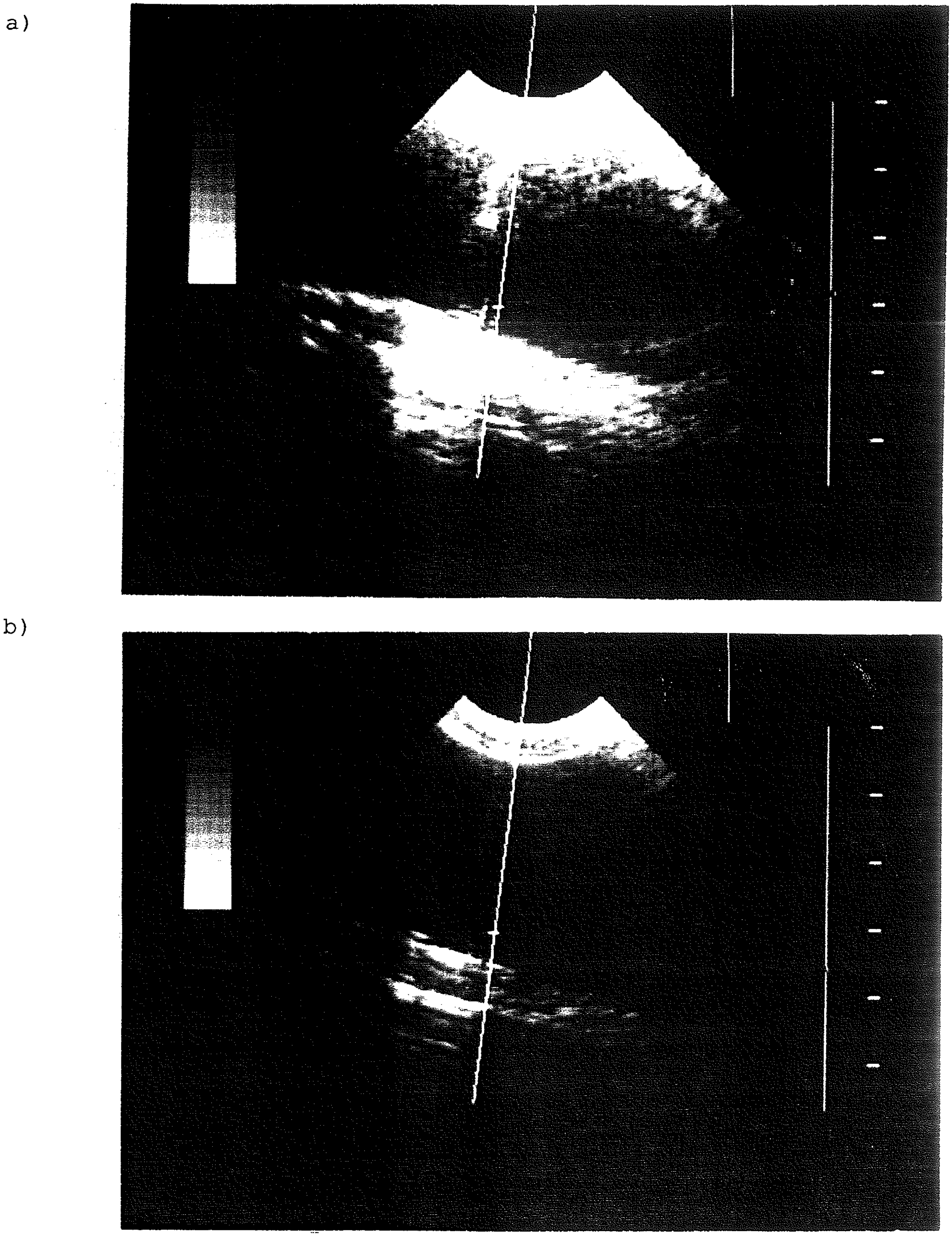


Figure 60 - Long-axis ultrasound image of the 6.5 mm vessel at a) maximum and b) minimum output power settings of the MK500 860C imager.

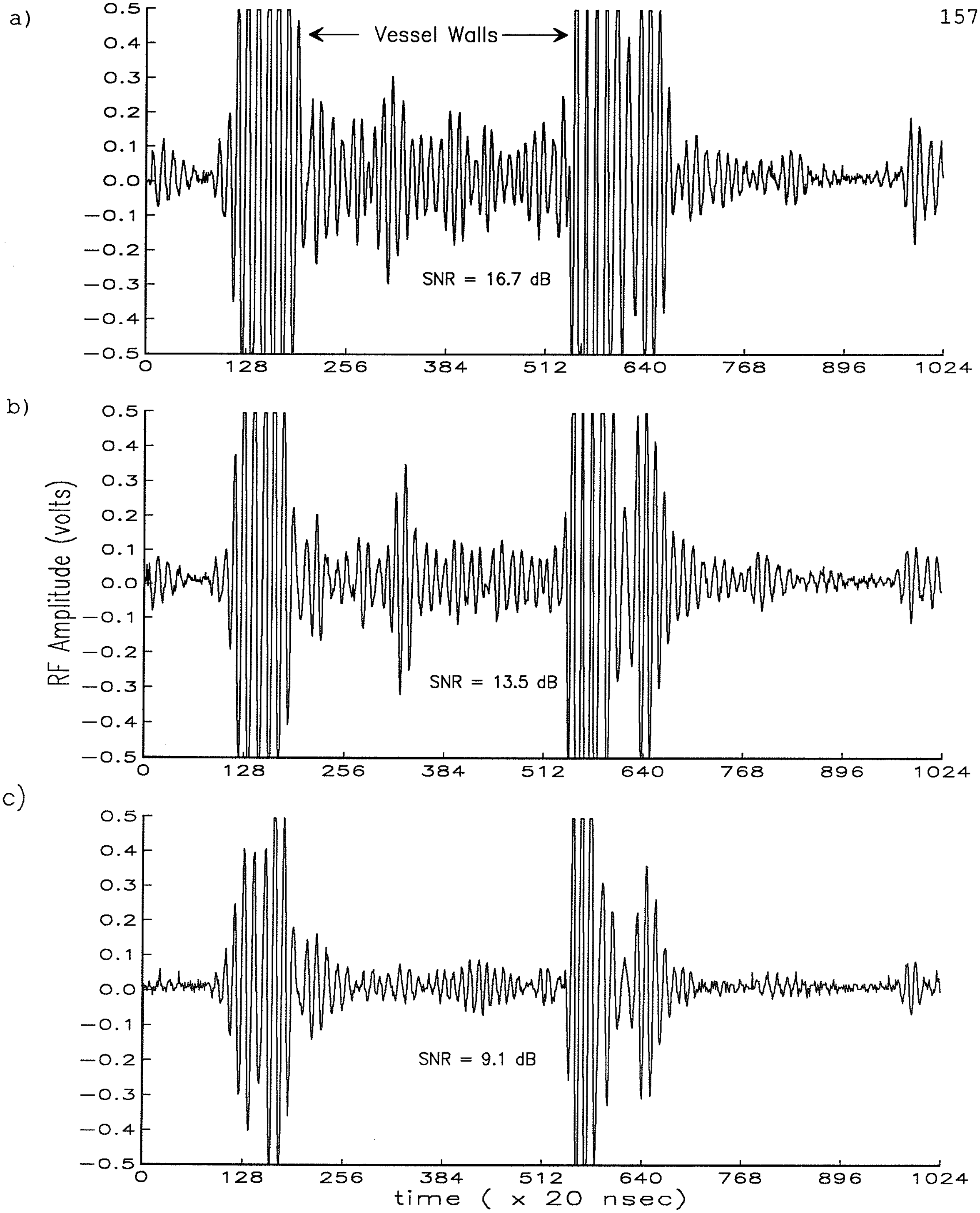


Figure 61 - RF signal digitized at different power output settings of the MK500 imager. a) 16.5 dB b) 12.0 dB c) 7.5 dB. The large amplitude sections are due to the vessel walls.

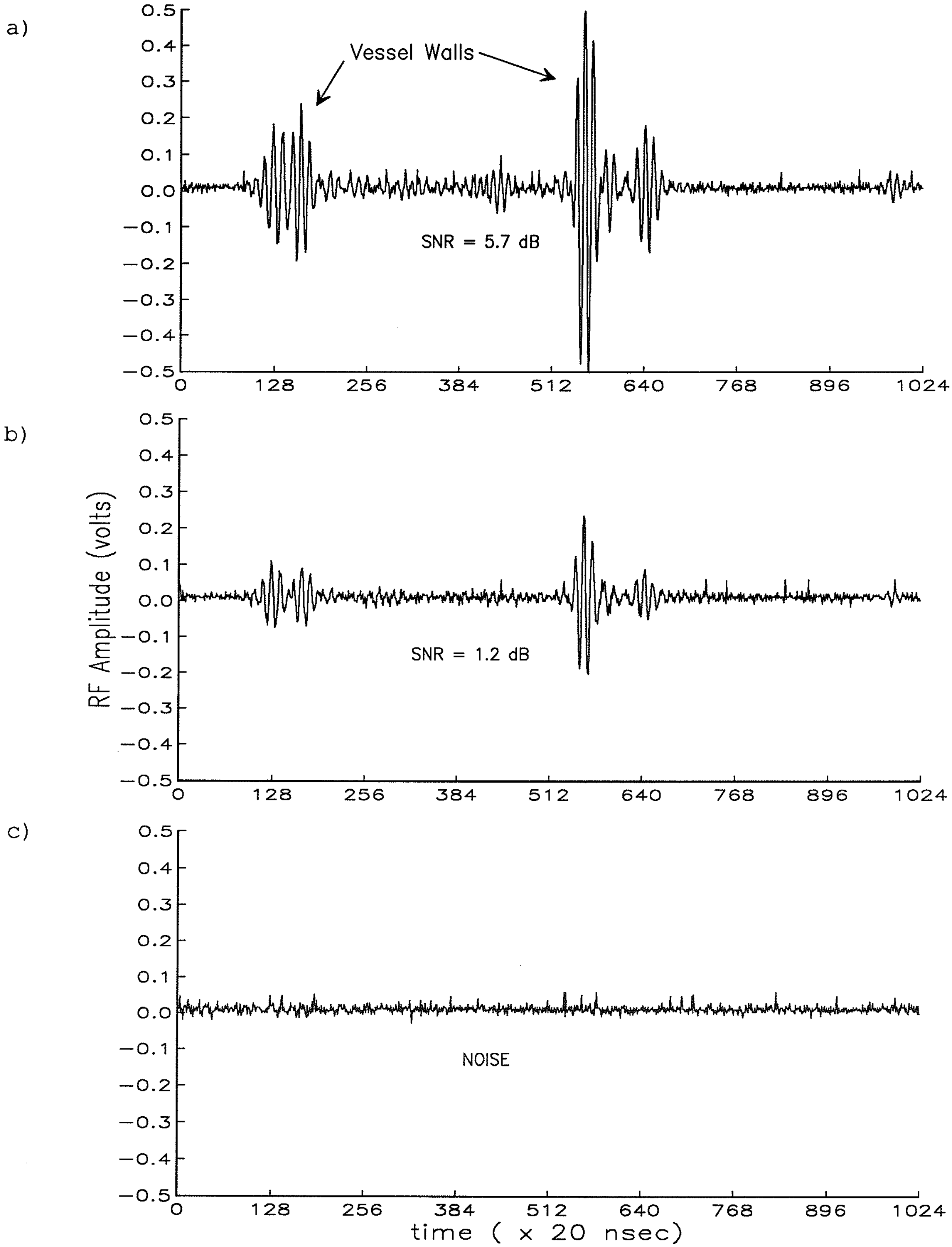


Figure 62 - RF signal digitized at different power output settings of the imager. a) 13.0 dB b) 0.0 dB c) Noise.

and 62 are very obvious since they produce a very large reflection. This corresponds to the very bright areas along the wiper at the cursor position in Figure 60. The signal from the vessel walls was clipping the A/D for power levels of 16.5, 12, and 7.5. This clipping was not severe enough to affect the correlation algorithm. The ultrasound reflected from fluid flow occurs between these large peaks. Figure 62c) also shows the RF noise of the system. The noise of the system was measured by removing the transducer from the water bath. This results in no transmitted ultrasound into the air and no reflections, and a black image. A section of the black image was digitized and any measured signal was considered to be the noise of the system. The signal-to-noise ratio for each power setting was estimated by calculating the time-averaged power over a 128-point segment of the echo, where the 128-point segment corresponded to an area of flow within the vessel (the same 128-pt region was used for each power setting echo). This power level was referenced to the noise power in the SNR calculation. Twenty consecutive measurements were made at the power settings of 16.5, 12.0, 7.5, 3.0, and 0.0 dB. The results of these experiments are listed in Table 6. Analysis of Table 6 indicates that reasonable results can be obtained with SNRs down to 9.1 dB, and possibly lower. At an SNR of 5.7 dB (Power = 3 column in Table 6), the error becomes large and negatively biased. The average error is -56.4%, and the standard deviation in the error is 10.5%. This standard deviation in the error is not significantly larger than that of measurements

Table 6: Measurements made at different power settings of the MK500.

Hydrodynamic flow rate = 111.3 ml/minute
Measurement angle = 74.6

	Power = 16.5 SNR = 16.7		Power = 12.0 SNR = 13.5		Power = 7.5 SNR = 9.1		Power = 3.0 SNR = 5.7		Power = 0.0 SNR = 1.2	
Measurement Number	Flow ml/min	Error %	Flow ml/min	Error %	Flow ml/min	Error %	Flow ml/min	Error %	Flow ml/min	Error %
1	119	+ 6.9	109	- 2.7	105	- 5.7	39.5	-64.6	Not Measurable	
2	118	+ 6.1	105	- 5.9	101	- 9.0	32.5	-70.8		
3	120	+ 8.1	123	+10.2	120	+ 7.5	48.6	-56.3		
4	117	+ 4.7	116	+ 4.2	107	- 3.7	53.8	-51.7		
5	121	+ 8.7	103	- 7.6	106	- 4.4	49.7	-55.4		
6	120	+ 7.8	125	+12.1	113	+ 1.7	53.7	-51.8		
7	123	+10.9	122	+ 9.8	91	-18.5	35.3	-68.3		
8	119	+ 6.6	101	- 9.5	103	- 7.5	33.8	-69.7		
9	105	- 5.5	126	+13.5	102	- 8.7	55.8	-49.9		
10	125	+12.4	107	- 4.2	104	- 6.7	56.9	-48.8		
11	116	+ 4.7	122	+ 9.5	98	-12.3	52.9	-49.1		
12	117	+ 4.9	98	-11.5	108	- 2.6	44.2	-52.4		
13	121	+ 8.4	106	- 4.9	100	-10.4	58.4	-60.3		
14	119	+ 6.9	118	+ 6.0	105	- 5.7	50.4	-47.6		
15	113	+ 1.9	102	- 8.3	117	+ 5.5	42.0	-54.7		
16	117	+ 5.5	119	+ 7.3	102	- 8.7	44.6	-62.3		
17	127	+14.5	102	- 8.5	112	+ 0.2	46.7	-60.0		
18	112	+ 0.7	115	+ 3.6	114	+ 2.2	82.5	-58.0		
19	121	+ 8.7	114	+ 2.9	105	- 5.7	44.4	-25.9		
20	114	+ 2.5	122	+ 9.4	105	- 5.7	32.5	-70.8		
	avg err = +6.3% std dev = 4.4		avg err = +1.3% std dev = 8.3		avg err = -4.9% std dev = 6.1		avg err = -56.4% std dev = 10.5			

made at the higher SNRs. At an SNR of 1.2 dB there was no discernible velocity.

The angle-corrected velocity vs. range plots for the measurements made at the five power levels are shown in Figure 63. The plots look approximately parabolic down to an SNR of 9.1 dB. At an SNR of 5.7 dB the plot still looks somewhat parabolic but the velocity values are very negatively biased. At the lowest power setting there is no parabolic shape, and there appears to be slight negative velocities. It is possible that the noise in the MK500 system may correlate in the negative direction. The noise may not have a major effect at higher SNRs, but may become significant when the signal approaches the level of the noise, as for the SNR = 5.7 dB case. The effect here is to produce a negative bias on the velocity estimate. When the signal is essentially noise, the small negative velocity values are present, as in the bottom plot of Figure 63. This negative bias is most likely due to the noise in the MK500 system. The correlation algorithm and UDA-RNC were tested by removing the RF signal of the MK500 from the A/D and feeding a separate random noise source to the input of the A/D. The system produced a flat zero-level velocity response for the separate noise source.

8.2.3 Measurements with attenuating and reflecting media

The dialysis tubing was surrounded by a natural wood sponge to examine the effects of an attenuating and reflective medium on

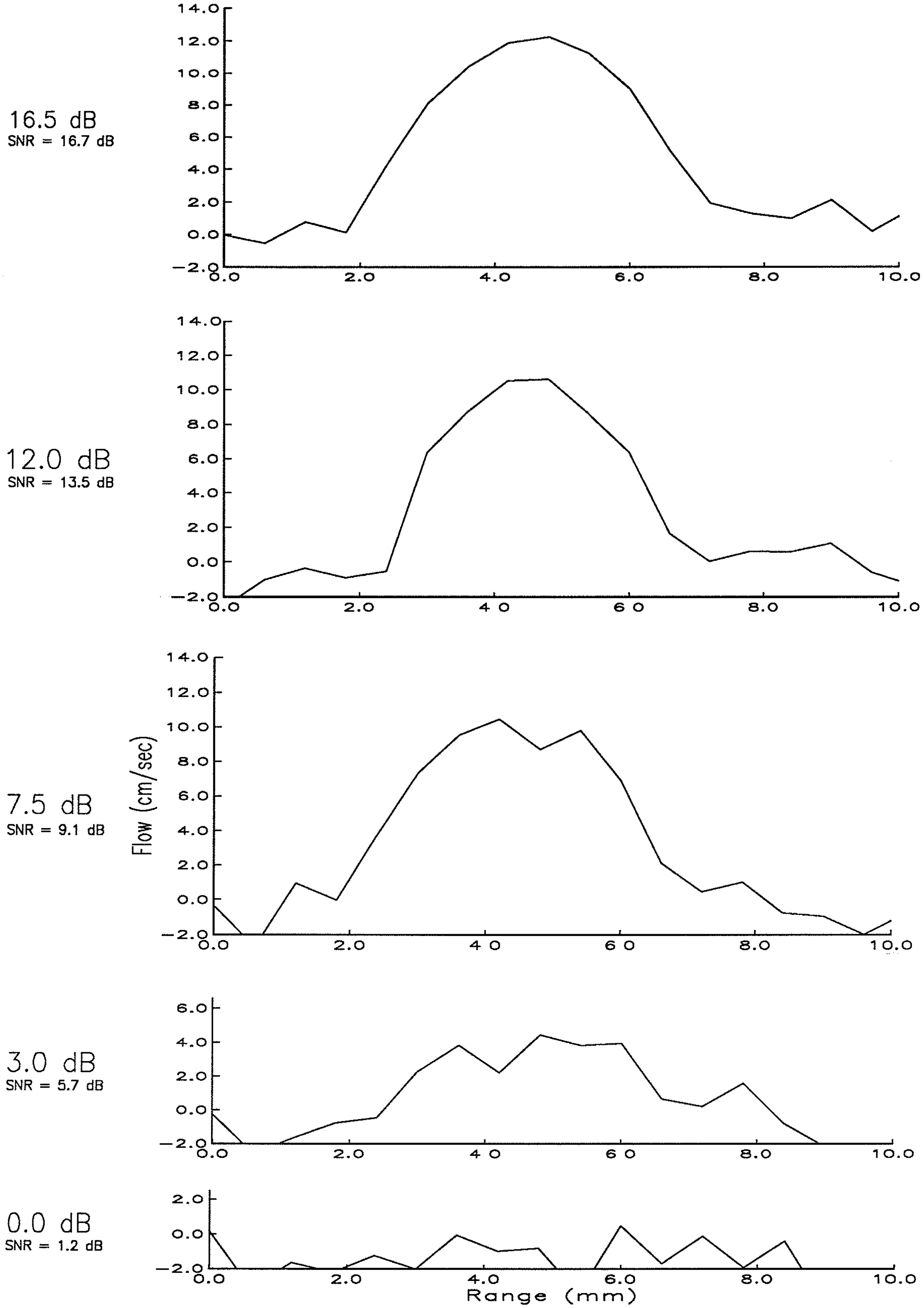


Figure 63 - Velocity profile across vessel at different power settings of the MK500. The hydrodynamic flow was kept constant as the power was changed.

the velocity estimates. Figure 64a) shows the long-axis ultrasound image of the sponge-surrounded vessel. The attenuation of the sponge was estimated by measuring the magnitude of the RF signal reflected from the vessel walls with and without the sponge present. The loss due to insertion of the sponge was measured to be approximately 9.8 dB. A number of experiments were done at different flow rates and angles with the sponge present. The results of these experiments are listed in Table 7. The output power of the MK500 was set at the maximum level for these measurements. The average error for the 296 ml/min $\angle 78^{\circ}$ measurements was -33.7%, -31.6% for the 177 ml/min $\angle 73^{\circ}$ measurements. These results are in accordance with the low SNR results obtained in the previous section. The 120 ml/min $\angle 65^{\circ}$ measurements had a much better average error, -6.0%. This may be due to the smaller measurement angle. The precision of the UTDC technique is dependent on both the SNR and the measurement angle. At this angle the negative effects of the decreased SNR due to the sponge may be offset by the higher measurement angle.

Figure 64b) shows the velocity vs. range plot for a measurement made at 78°. In addition to the negative bias, negative-going spikes sometimes appeared in the velocity vs. range plots with the sponge present. These spikes may be due to multiple reflections of the ultrasound signal within the sponge, which could cause cancellations of the signal at certain points within the vessel. It is also possible that the stationary echoes

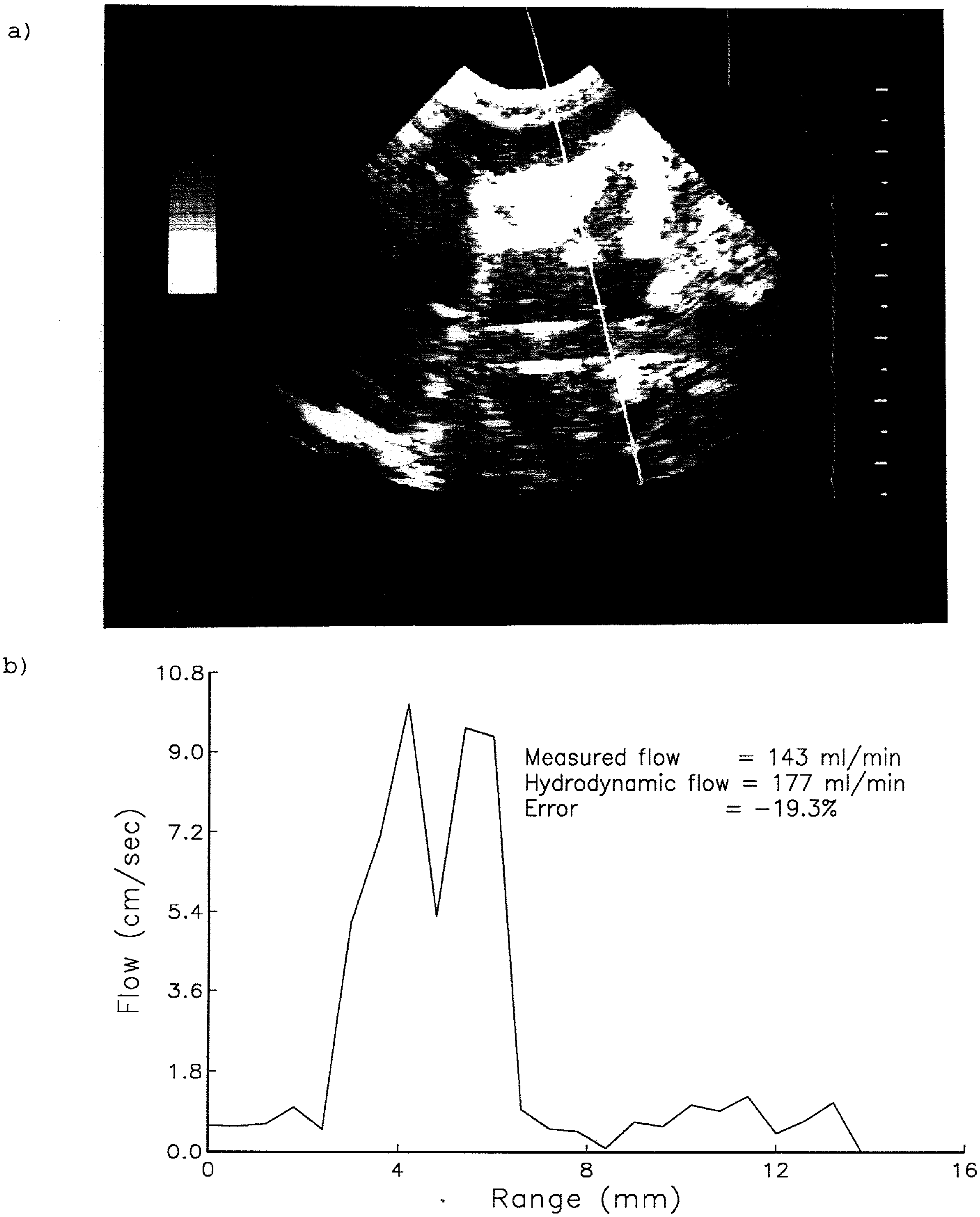


Figure 64 - a) Long-axis ultrasound image of a 6.5 mm vessel surrounded by natural wood sponge. Estimated angle between transducer and vessel is 78 deg. b) Velocity profile across vessel.

Table 7: Measurements made with attenuating sponge surrounding vessel.

	Flow = 296 ml/min Angle = 78 deg		Flow = 177 ml/min Angle = 73 deg		Flow = 120 ml/min Angle = 65 deg	
Measurement number	Measured Flow ml/min	Error %	Measured Flow ml/min	Error %	Measured Flow ml/min	Error %
1	196	-33.8	108	-38.8	114	- 5.0
2	259	-12.5	158	-10.9	104	-13.5
3	179	-39.4	122	-31.3	103	-14.5
4	197	-33.3	101	-43.2	104	-13.5
5	144	-51.4	152	-14.1	126	+ 5.4
6	232	-21.6	88	-50.2	132	+10.0
7	229	-22.7	136	-23.3	129	+ 7.7
8	160	-45.9	123	-30.6	109	- 9.3
9	237	-19.9	82	-53.9	99	-17.6
10	200	-32.9	81	-54.1	125	+ 4.1
11	204	-30.9	108	-39.3	108	-10.4
12	206	-30.3	143	-19.3	137	+14.6
13	145	-51.0	113	-36.3	98	-18.8
14	138	-50.4	182	+ 3.1	111	- 7.4
15	170	-42.7	142	-19.9	114	- 5.4
16	216	-26.9	138	-22.0	100	-16.9
17	253	-14.5	86	-51.7	89	-25.5
18	203	-31.4	87	-50.4	131	+ 9.5
19	149	-49.6	98	-44.6	122	+ 1.6
20	196	-33.7	179	+ 1.3	101	-16.0
	avg error = -33.7% std dev = 12.3		avg error = -31.6% std dev = 17.8		avg error = -6.0% std dev = 11.5	

from the sponge may be causing a negative bias in the velocity estimates. If this is the case a stationary echo canceler may have to be added to the system.

8.2.4 Effects of different numbers of echoes and Δ_{max}

The speed of the real-time system depends directly on the number of echoes used and Δ_{max}. To analyze the effects of these parameters on the speed and accuracy of the UTDC measurement method, three sets of measurements were performed: 1) using 210 echoes and a Δ_{max}=10; 2) 20 echoes and a Δ_{max}=10, and 3) 210 echoes and a Δ_{max}=1. Twenty experiments were performed at each of these parameter settings, and are listed in Table 8. The accuracy of each parameter setting is very similar. The average accuracy for Δ_{max}=10 with 210 echoes was -3.92%, for Δ_{max}=10 with 10 echoes was +1.08%. and for Δ_{max}=1 with 210 echoes was +3.0%. The best standard deviation was 2.87 with Δ_{max}=10 and 210 echoes. This is to be expected since for this parameter setting the most averaging takes place. The standard deviations for Δ_{max}=10 and 20 echoes was 6.29 and for Δ_{max}=1 and 210 echoes was 10.0. These standard deviations are somewhat higher but are still acceptable. As stated in Chapter 8, Section 1.5, the highest speed was obtained at the Δ_{max}=1 and 210 echo setting. This parameter was chosen as the default setting for the real-time system.

Table 8: Measurements made with different acquisition parameters. The hydrodynamic flow was 181 ml/min and the angle was 73 degrees.

	Delta Max = 10 Echoes = 210		Delta Max = 10 Echoes = 20		Delta Max = 1 Echoes = 210	
Measurement number	Measured Flow ml/min	Error %	Measured Flow ml/min	Error %	Measured Flow ml/min	Error %
1	184	+ 1.2	173	+ 4.7	173	- 4.9
2	169	- 6.6	194	+ 7.2	189	+ 4.0
3	165	- 9.2	183	+ 0.6	190	+ 4.5
4	171	- 6.0	169	- 6.8	168	- 7.2
5	176	- 3.2	188	+ 3.9	180	- 1.0
6	172	- 5.3	192	+ 5.7	183	+ 1.0
7	171	- 5.6	162	-10.9	163	-10.2
8	169	- 6.7	184	+ 1.3	228	+25.6
9	182	+ 0.3	177	- 2.7	170	- 6.3
10	176	- 3.2	192	+ 5.5	233	+28.3
11	169	- 6.7	177	- 2.4	185	+ 2.2
12	183	+ 0.7	195	+ 7.2	185	+ 2.2
13	175	- 3.6	185	+ 1.8	165	- 9.1
14	183	+ 0.6	192	+ 5.5	183	+ 0.7
15	177	- 2.4	192	+ 5.5	182	+ 0.2
16	172	- 5.0	182	+ 1.7	189	+ 4.2
17	177	- 2.7	189	+ 3.9	190	+ 4.6
18	175	- 3.6	160	-11.6	174	- 4.0
19	171	- 6.0	203	+11.6	180	- 0.9
20	172	- 5.4	184	+ 1.3	185	+ 2.2
	avg error =	- 3.9%	avg error =	+ 1.6%	avg error =	+ 1.8%
	std dev =	2.9	std dev =	6.0	std dev =	9.7

8.2.5 Multiple vessel experiments

Blood vessels frequently lie next to each other in living beings, so experiments were performed with two vessels in the ultrasound beam. Chapter 5, Section 2.3 described an experiment where the vessels were placed next to each other with respect to the transducer (see Figure 39). A more interesting experiment is to orient the transducer such that one vessel lies in front of the other with respect to the transducer, as shown in Figure 65. The vessels were placed such that walls were touching. This would ensure that both vessels would fall within the 15 mm sampling window at the cursor position. In this orientation, echoes reflected from the flow in the rear vessel will also pass through flow in the front vessel. To see if the presence of the front vessel has an effect on velocity measurement in the rear vessel, experiments were set up as follows. First, one-dimensional measurements were made with both vessels present in the ultrasound beam. The ultrasound image of this measurement is shown in Figure 66. The volumetric flow through each vessel was calculated as described in Chapter 6, Section 2. The sum of the volumetric flow through both vessels was compared to the hydrodynamic flow. Afterwards, one-dimensional measurements were made in each vessel alone; that is, measurements were made in the rear vessel after moving the front vessel out of the ultrasound beam, and in the front vessel after moving the rear vessel out of the beam (the flow through each vessel was unchanged). The flow

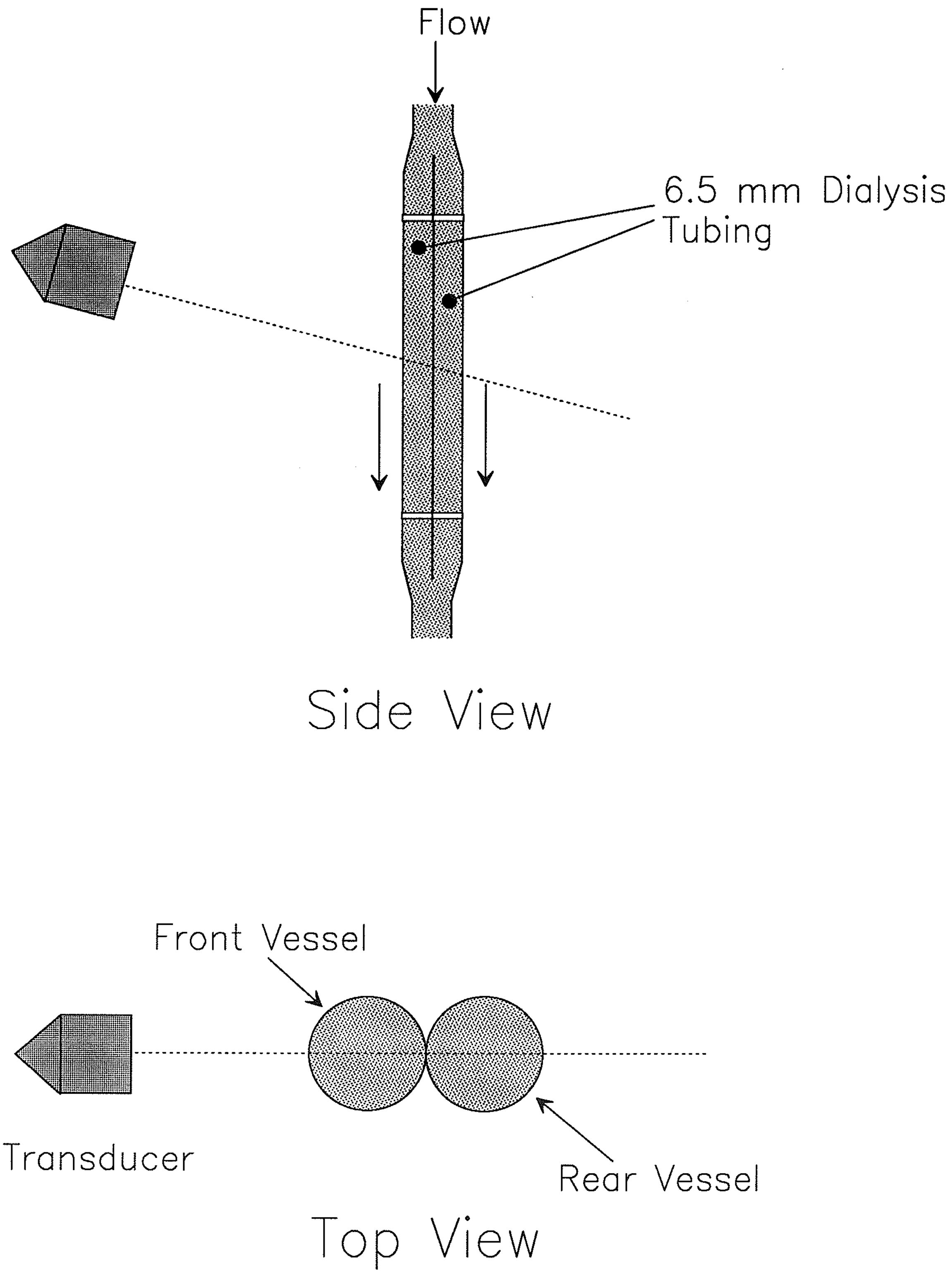


Figure 65 - The transducer is oriented such that one vessel is behind the other. A one-dimensional scan will intersect both vessels.

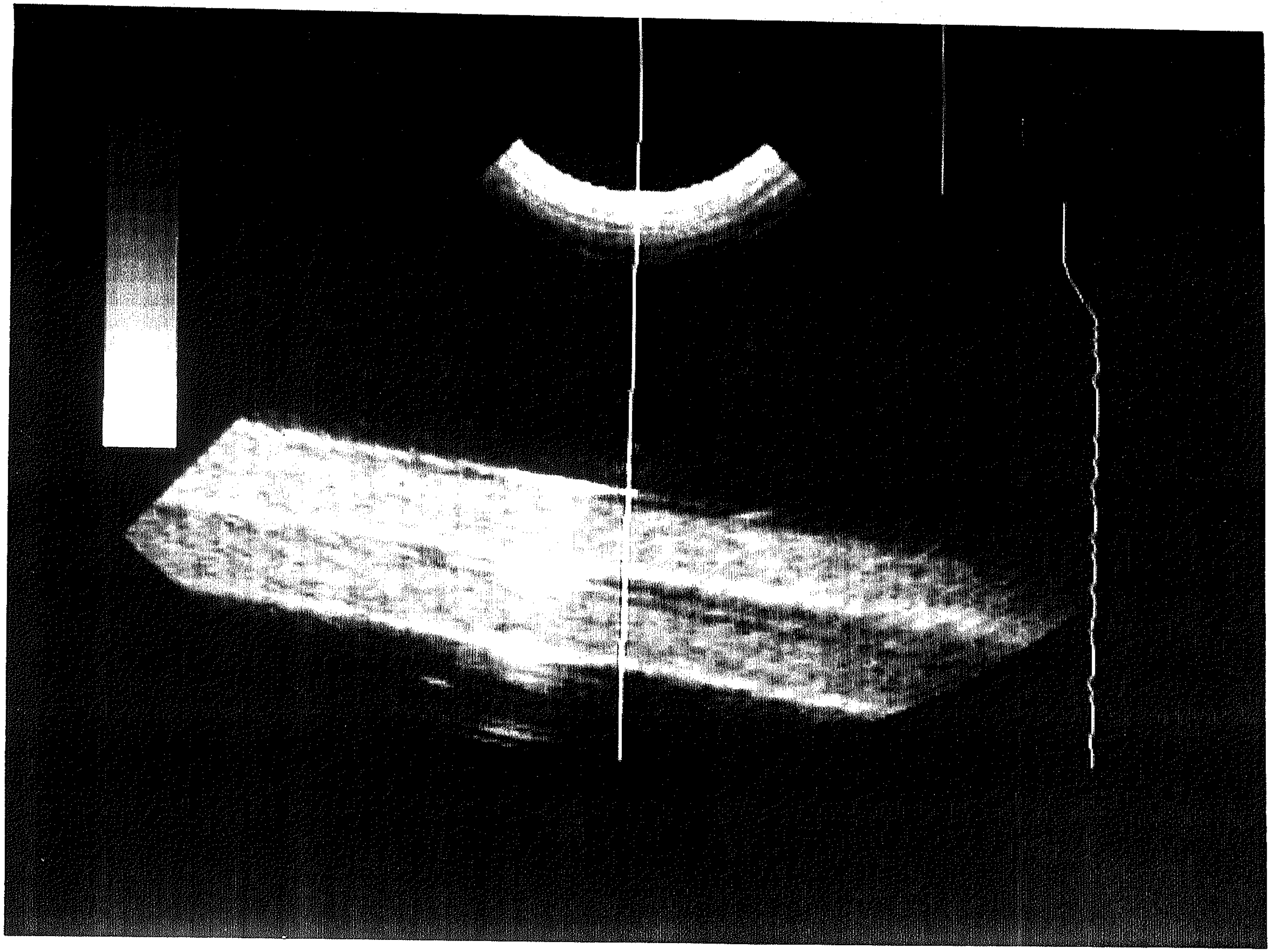


Figure 66 - Long-axis ultrasound image of two 6.5 mm vessels placed next to each other. One vessel is behind the other with respect to the transducer field of view.

measured in each vessel alone was then compared to the flow measured in each with both vessels present.

Velocity profiles for the three configurations are shown in Figure 67. Figure 67a) shows the velocity profile for both vessels in the ultrasound beam, Figure 67b) shows the profile in the front vessel with the rear moved and 67c) shows the profile in the rear vessel with the front vessel moved. These profiles show that the flow in both vessels is approximately parabolic, and that the flow does not appear to split evenly between the two vessels. It is also possible that the one-dimensional scan line did not pass through the center of both vessels. If the front vessel was slightly off center (with respect to the beam), it would have a smaller velocity profile.

Twenty consecutive measurements were made in each configuration, and the results are listed in Table 9. Columns A through D in Table 9 list measurements made with both vessels in the ultrasound beam. Column A is the volumetric flow estimated in the front vessel, column B is the flow in the rear vessel, and column C is the sum of the two. Column D is error in the sum of the flow through both vessels referenced to the hydrodynamic flow. Columns E through F in Table 9 are the same measurements made except that the front vessel was moved out of the beam when measurements in the rear vessel were made and vice versa. The flow through the vessels was not changed. Column E is the flow measured in the front vessel alone, and column F is the flow

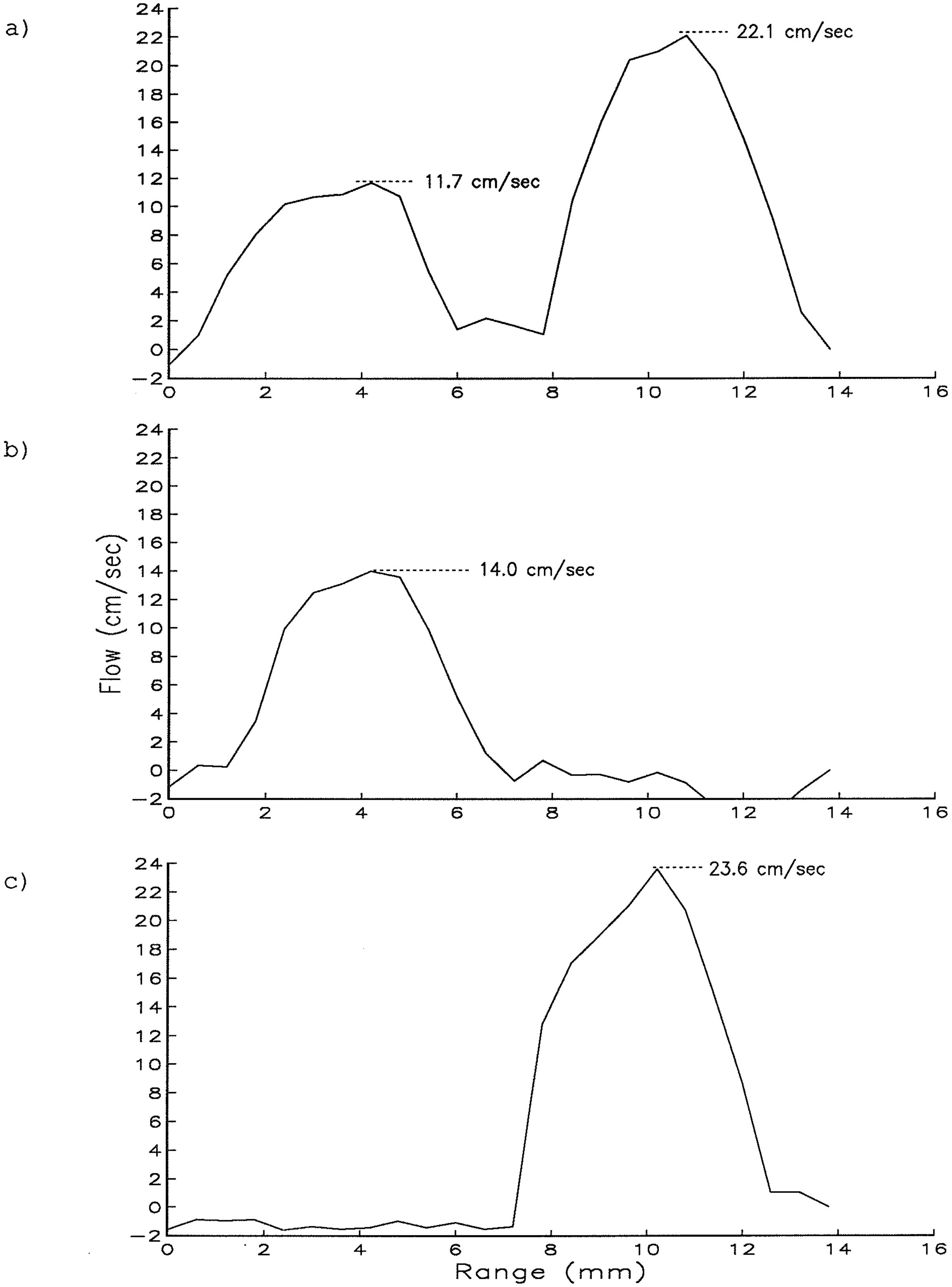


Figure 67 - a) Velocity profile of two vessels with one vessel behind the other. b) Profile of front vessel with rear vessel moved out of the transducer field. c) Profile of rear vessel with front moved out of transducer field.

Table 9: Measurements made with two vessels in the beam. The total hydrodynamic flow rate was 322.4 ml/min

	Both vessels in beam during measurement				Only one vessel in beam during measurement			
	A	B	C	D	E	F	G	H
Measurement Number	Front Flow ml/min	Rear Flow ml/min	Total Flow ml/min	Error %	Front Flow ml/min	Rear Flow ml/min	Total Flow ml/min	Error %
1	89	141	230	-28.9	95	135	230	-28.5
2	109	175	284	-11.9	101	189	290	- 9.9
3	110	140	250	-22.4	141	181	322	0.0
4	107	164	271	-15.8	115	140	255	-20.8
5	140	197	227	-29.5	132	169	301	- 6.5
6	111	157	268	-16.7	104	161	265	-17.7
7	113	171	284	-11.8	152	146	298	- 7.4
8	134	165	299	- 7.1	143	140	283	-12.1
9	108	208	316	- 1.9	85	134	219	-31.9
10	94	184	278	-13.7	113	176	289	-10.2
11	99	171	270	-16.1	104	125	229	-28.9
12	96	216	312	- 3.1	109	141	250	-22.3
13	107	163	270	-16.1	144	118	262	-18.6
14	110	185	295	- 8.4	134	204	338	+ 5.9
15	84	158	242	-24.8	91	133	224	-30.4
16	110	176	286	-11.1	136	139	275	-14.5
17	110	176	286	-11.1	92	135	227	-29.5
18	133	195	328	+ 1.9	122	154	276	-14.2
19	119	189	304	- 5.6	157	150	307	- 4.6
20	109	175	284	-11.8	106	131	237	-26.4
Average	110	175	279	-13.1%	119	150	268	-17.5%
Std Dev	14	20	27	9.0	22	23	35	10.2

measured in the rear vessel alone. Column G is the sum of the two, which should equal the hydrodynamic flow, and column H is the error in the flow through both vessels as referenced to the hydrodynamic flow.

The average error in both sets of measurements was similar. The average error was -13.1% for both vessels in the beam, and -17.5% for only one vessel in the beam at a time. The standard deviation in the error was also very similar, 9.0% and 10.2%, respectively. One possible reason for this negative average error is that it was difficult to orient the transducer such that the beam passed through the exact center of both vessels. If the beam were passing through the center of one vessel and not through the center of the other, the volumetric flow would be underestimated in one and the overall result would have a negative average error. The average flow measured in the front vessel for both configurations (columns A and E) was also very similar. The average flow was 110 ml/min with both vessels in the beam and 119 ml/min with only one vessel within the beam. There was a larger discrepancy for flow measured in the rear vessel (columns B and F). With both vessels in the beam, the average measured flow was 175 ml/min, and for only one vessel in the beam, the average measured flow was 150 ml/min. Part of the reason for the larger difference may be due to the presence of the front vessel. Individual echoes reflected from the rear vessel and passing through the front vessel will be affected differently due to scatterer motion in the front vessel. It is also possible that when the front vessel was moved out of the beam, the rear vessel was also moved slightly and the ultrasound beam was not passing through the center of the vessel during the measurements. This would account for a lower volumetric flow estimate. This is most likely the cause of the discrepancy, since the average flow with both vessels in the beam was better than when measured individually (-13.1% as opposed to -17.5%).

These multiple vessel measurements were limited in that the volumetric flow through both vessels could not be changed independently and the volumetric flow through each vessel could not be independently measured (the hydrodynamic flow measurement provided only the sum of the flow through the vessels). A more thorough investigation of multiple vessel flow would require modification of the blood flow phantom such that the volumetric flow through each vessel could be independently set and measured. Vessels with flow in opposite directions would be particularly interesting, since the human body frequently has arteries and veins laying next to each other.

8.2.6 Measurements in a 1.5 mm diameter tube

Most of the experiments performed in the blood flow phantom have been made with 6.5 mm dialysis tubing. Blood vessels can, of course, be much smaller in the human body. To simulate smaller vessels, the blood flow phantom was modified to allow 2.3 mm (1.5

mm inner diameter) polyethylene tubing to be used, as shown in Figure 68. The ultrasound image of the smaller tube is shown in Figure 69. This tubing is somewhat rigid and has a much thicker wall than dialysis tubing. The resistance to flow was much greater for the 1.5 mm tube than for the usual 6.5 mm dialysis tube; thus, a maximum hydrodynamic flow rate of only 20.5 ml/min was obtainable. In addition, the flow was not smooth but dripped out of the bottom end of the tube (see Figure 68); hence, it was somewhat pulsatile in nature. Figure 70 shows the velocity profile for flow through the tubing using low resolution (0.6 mm/data point) and high resolution (0.15 mm/data point). At low resolution, only a single point appears. High resolution shows a much more realistic plot of flow in the vessel. At the measurement instant, the flow appears to be plug flow. The use of high resolution will be necessary for measuring flow in smaller vessels, since low resolution may miss a very small vessel completely.

8.3 *In Vivo* Measurements

Measurements have also been made in the carotid artery of a human subject (the author of this thesis). The carotid artery was located on the ultrasound image and the wiper and cursor were set accordingly. The ultrasound image of the carotid artery is shown in Figure 71. This image was shown to a sonographer [38] who indicated with a high level of certainty that this was the carotid

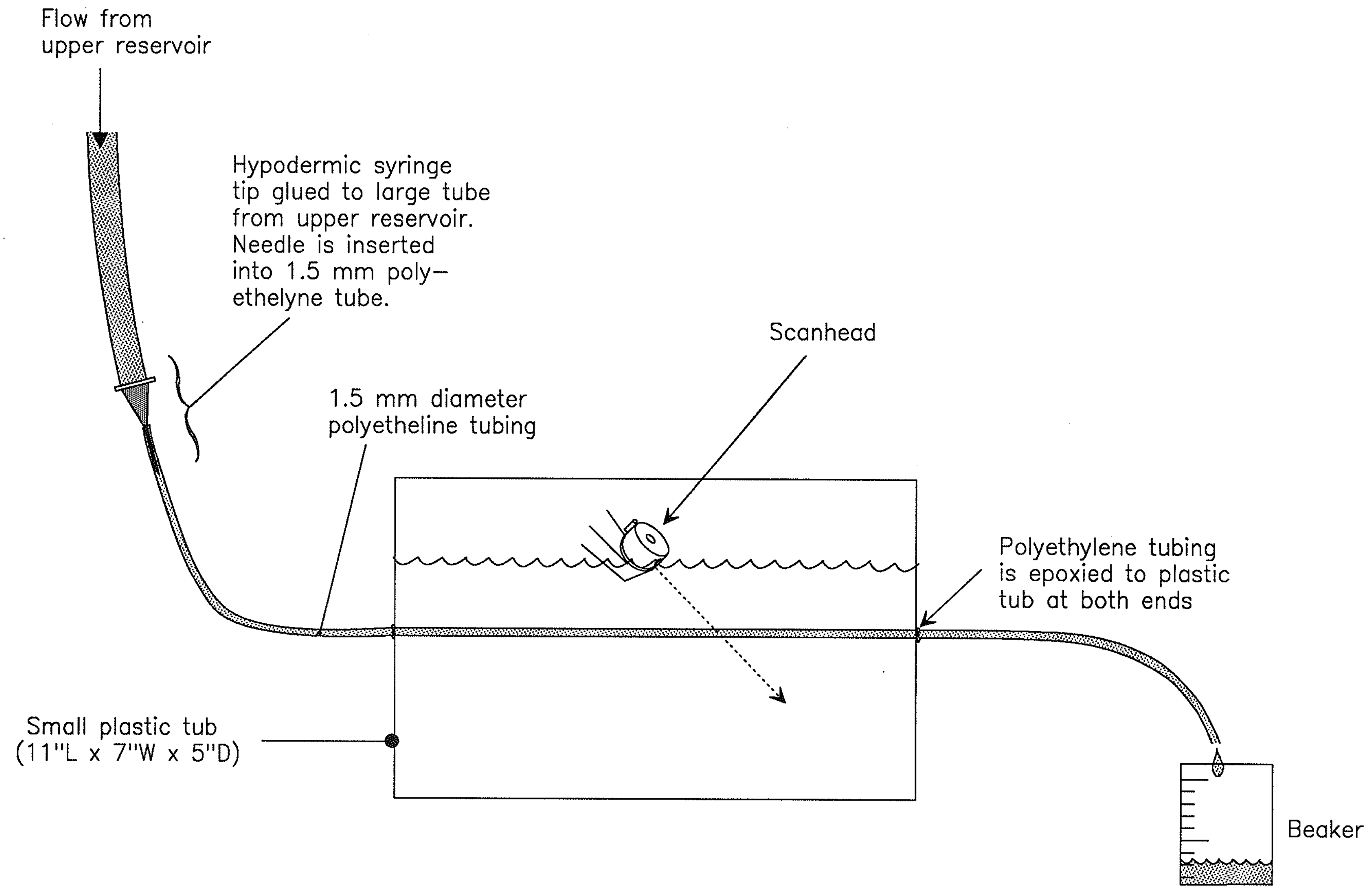


Figure 68 - Modification of blood flow phantom to allow measurements of flow within a 1.5 mm inner diameter polyethylene tube.

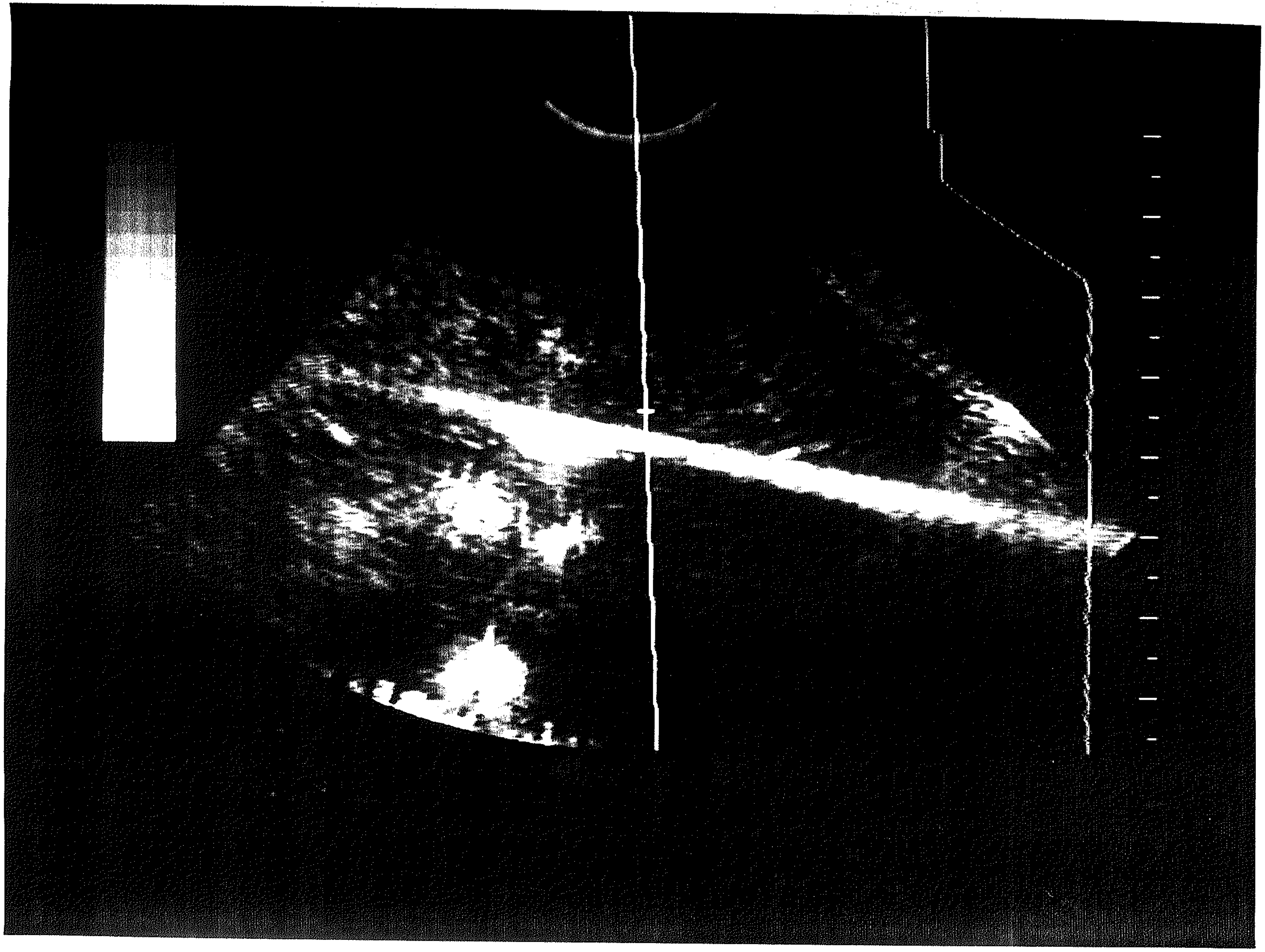


Figure 69 - Long-axis ultrasound image of 1.5 mm inner diameter polyethylene tube.

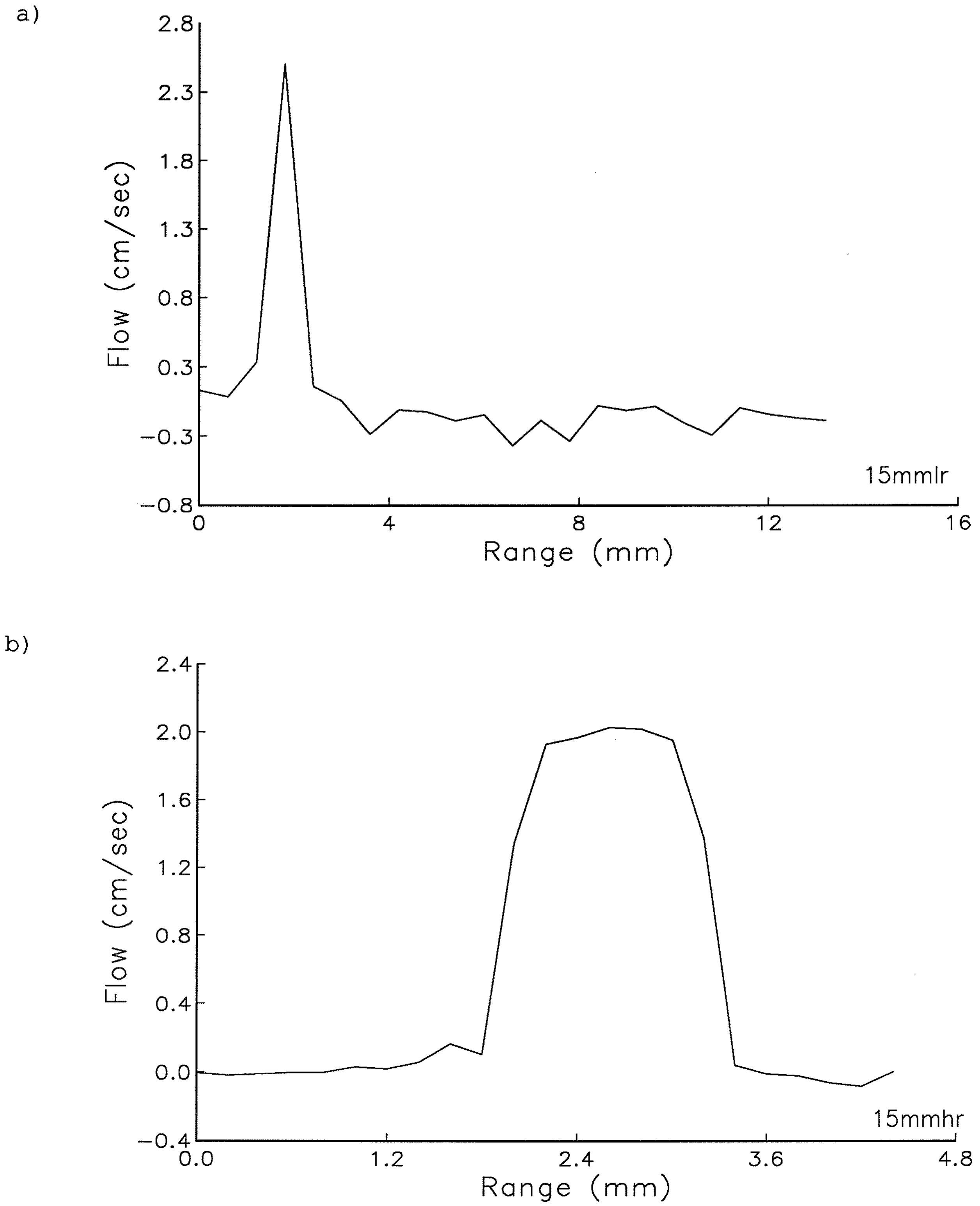


Figure 70 - Velocity profiles measured in a 1.5 mm diameter tube at a) low resolution (0.6 mm/data pt) and b) high resolution (0.125 mm/data pt). Transducer angle = 70°, hydrodynamic flow = 20.5 ml/m.

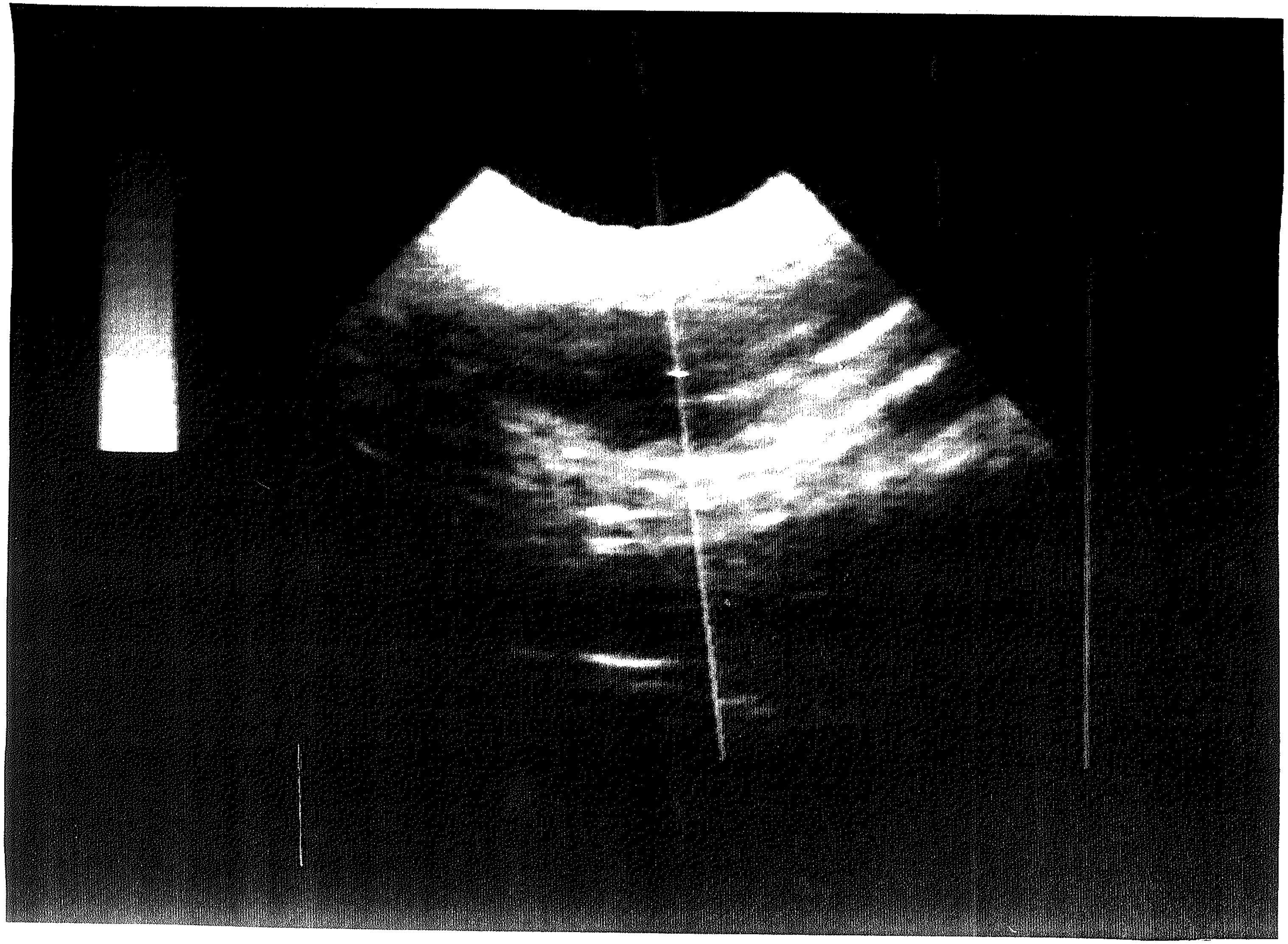


Figure 71 - Long-axis ultrasound image of the human carotid artery. The wiper intersects the artery just past the cursor where the artery is curving. The angle θ was estimated to be 85 degrees.

artery. The artery is the curving structure just past the cursor and runs longitudinally from the center of the image out to the right-hand side of the image. The walls at certain positions are bright white. The positioning of the wiper angle was somewhat tricky because of the high velocities in the artery. From Figure 71, the angle was estimated at approximately 85°, which fixed the axial component of velocity within the measurable range (less than 10 cm/sec). As stated earlier, the axial velocity is directly proportional to $\cos(\theta)$. The cosine function changes rapidly at very high angles, and if the angle is too great, the axial velocity will be too small and not measurable, and if it is too small, the axial velocities will be too great to be measured. For example, if the actual velocity is 40 cm/sec, the measurement angle θ must be between 75° and 85° for the axial component of flow to fall between 10 and 1 cm/sec, respectively. The angle-corrected velocity vs. range plot for flow within the carotid artery is shown in Figure 72. These velocity profiles were also shown to the sonographer, who indicated that the vessel widths and velocity levels are in the range of those found in the carotid artery. The top plot in Figure 72 shows the velocity plots as calculated in real time (210 echoes with $\Delta_{max}=1$).

The RF echoes were also saved and later processed with different Δ_{max} and number of echoes parameters. The middle plot in Figure 72 shows the velocity calculated with $\Delta_{max}=10$ and 20 echoes, and the bottom plot with $\Delta_{max}=10$ with 210 echoes. All three

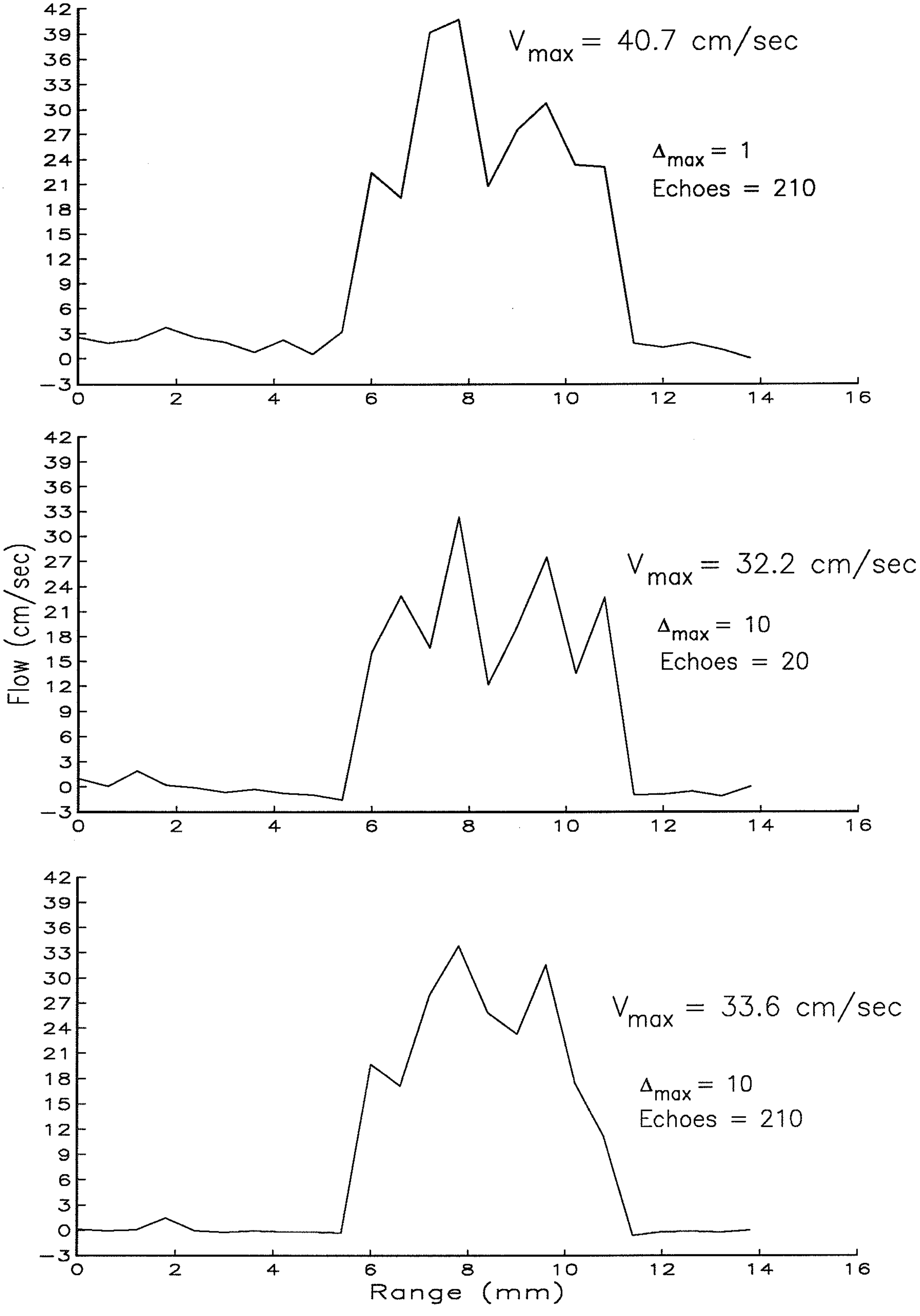


Figure 72 - Velocity vs. range measured in the human carotid artery calculated with different Δ_{max} and number of echoes. The top plot was calculated in real time. The RF echo was saved to disk and the bottom two calculated later.

processing parameters estimate the location of the vessel walls to be the same. The location of the vessel (with respect to the cursor) and the vessel wall width are in good agreement with the vessel location and width in the ultrasound image. The shape of the velocity plot appears somewhat different for each set of parameters. Without knowing the true velocity profile, it is difficult to say which one is the most correct (the subject was not too keen on having an electromagnetic flowmeter probe invasively mounted on his carotid artery). However, the bottom plot of Figure 72 (Δ_{max}=10, 210 echoes) has the most averaging (1,800 echo pairs) and also looks the "smoothest." The real-time plot (top, Δ_{max}=1, 210 echoes, 209 echo pairs) actually does not look that much different from the bottom plot, except that the velocity peak at a distance of 8 cm is 7 cm/sec higher. Both the top and bottom plots use the same number of echoes for the Δ=1 echo spacing; the difference between the two plots is that the bottom plot also used weighted-averages in the Δ=2 through Δ=10 echo spacing values. The effect of averaging on the other Δ values thus brings down the large velocity peak in the top plot. Theoretically, the lower Δ values will estimate higher velocities better, and higher Δ will estimate slower velocities better. Thus the bottom plot should be more accurate over the entire velocity range. The middle plot (Δ_{max}=10 with 20 echoes for 90 echo pairs) most likely gives the least probable profile since only 9 echo pairs are used for each Δ value.

It is also possible that the surrounding tissues may be responsible for some of the sharp peaks and dips in the velocity profiles. The experiments with the sponge around the vessel showed that this spiking effect can be expected if there is an attenuating and scattering medium around the vessel. Future experiments should be conducted with a stationary-echo cancelling feature incorporated into the system.

8.4 Conclusions

The experiments in the blood flow phantom system have shown that the accuracy of the real-time system is the same as for the previous and the temporary systems. The system used 210 echoes and a Δ_{max}=1 for the real-time measurements, and can produce good results with an SNR as low as 9.1 dB. Multiple vessel experiments show that measurement of flow in adjacent vessels is possible, although more experiments with a more flexible blood flow phantom system should be made. Experiments with an attenuating and scattering medium indicate a stationary echo cancelling feature should be incorporated into the system. *In vivo* experiments have shown that the real-time system works within living beings.

The major limitation of the real-time system is the 1 kHz PRF rate, which could not be changed. This limited the measurable axial velocity to less that 10 cm/sec and required the use of a very large angle for measuring high flow rates. The use of a large angle reduces the precision of the UTDC technique. A future system should use a commercial imager with a variable PRF rate.

Another limitation of the current system is that it can only measure flow in one direction. The correlation algorithm should be modified such that flow in both directions can be measured without changing the orientation of the transducer.

The speed of the system can also be increased considerably with a number of future modifications. Currently, the raw correlations as described by Equation (36) are done in hardware, but the actual correlation algorithm (finding the location of the correlation coefficient) and velocity calculation are done in software. The correlation algorithm would run much faster if the entire correlation process were done in hardware. Also, the flowmeter program should be ported to the latest version of Metaware High-C 386. This will utilize the capabilities of the COMPAQ 386 to the fullest instead of running it as if it were only a fast 286 machine.

REFERENCES

[1] D.W. Baker, "Pulsed ultrasonic Doppler blood flow sensing," *IEEE Transactions on Sonics and Ultrasonics*, vol. SU-17, pp. 170-185, July 1970.

[2] D.J. Cathignol, C. Fourcade, and J. Chapelon., "Transcutaneous blood flow measurements using pseudorandom noise," *IEEE Transactions on Biomedical Engineering*, vol. BME-27, pp. 30-36, January 1980.

[3] P.N.T. Wells, *Biomedical Ultrasonics*. New York: Academic Press Inc., 1977.

[4] J.G. Webster, *Medical Instrumentation: Application and Design*. Boston: Houghton Mifflin Co., 1980.

[5] M.D. Olinger, "Ultrasonic blood flow imaging using correlation processing," Ph.D. dissertation, Dept. of Electrical Engineering and System Science, Michigan State University, East Lansing, MI, 1981.

[6] P.M. Embree and W.D. O'Brien, Jr., "Pulsed Doppler accuracy assessment due to frequency-dependent attenuation and Rayleigh scattering error sources," *IEEE Transactions on Biomedical Engineering*, vol. 37, no. 3, pp. 322-326, March 1990.

[7] S.G. Foster, "Pulsed ultrasonic flowmeter employing time domain methods," Ph.D. dissertation, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1981.

[8] S.G. Foster, P.M. Embree, and W.D. O'Brien Jr., "Flow velocity profile via time domain correlation: error analysis and computer simulation," *IEEE Transactons on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 37, no. 3., pp. 164-175, May 1990.

[9] P.M. Embree, "The accurate ultrasonic measurement of the volume flow of blood by time domain correlation," Ph.D. dissertation, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1986.

[10] P.M. Embree, "Volumetric bood flow via time-domain correlation: experimental verification," *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 37, no. 4, pp. 176-189, May 1990.

[11] I.A. Hein, V. Suorsa, J. Zachary, R. Fish, J. Chen, W.K. Jenkins, and W.D. O'Brien Jr., "Accurate and precise measurement of blood flow using ultrasound time-domain correlation," *IEEE 1989 Ultrasonics Symposium Proceedings*, pp. 881-886, Montreal, Canada, October 1989.

[12] I.A. Hein and W.D. O'Brien Jr., "Volumetric measurement of pulsatile flow via ultrasound time-domain correlation," *Journal of Cardiovascular Technology*, vol. 8, no. 4, pp. 339-348, 1989.

[13] D.G. Childers, D.P. Skinner, and R.C. Kemerait, "The cepstrum: a guide to processing," *Proceedings of the IEEE*, vol. 65, no.10, pp. 1428-1443, October 1977.

[14] R. Kuc, K. Haghkerdar, and M. O'Donnell, "Presence of cepstral peak in random reflected ultrasound signals," *Ultrasonic Imaging*, vol. 8, no. 3, pp. 196-212, July 1986.

[15] M.C. Rorke, F.L. Lizzi, D.L. King, E.J. Feleppa, M. Yaremko, and P. Wai, "Use of cepstrum analysis and cepstral imaging in characterizing liver tissue," *Ultrasonic Imaging*, vol. 7, no, 1, p. 90, January 1985.

[16] A. Nowicki and A. Marciniak, "Detection of wall vibrations by means of cepstrum analysis," *Ultrasonic Imaging*, vol. 11, pp. 273-182, 1989.

[17] T. Powalowski, "Ultrasonic system for noninvasive measurement of hemodynamic parameters of human arterial-vascular system," *Archives of Acoustics*, vol. 13, no. 1-2, pp. 89-108, 1988.

[18] R.J. Dickinson and C.R. Hill, "Measurement of soft tissue motion using correlation between A-scans," *Ultrasound in Medicine and Biology*, vol. 8, no. 3, pp. 263-271, 1982.

[19] M. Tristam, D.C. Barbosa, D.O. Cosgrove, J.C. Bamber, and C.R. Hill, "Application of Fourier analysis to clinical study of patterns of tissue movement," *Ultrasound in Medicine and Biology*, vol. 14, no. 8, pp. 695-707, 1988.

[20] O. Bonnefous, "Statistical analysis and time domain correlation processes applied to velocity measurement," *IEEE 1989 Ultrasonics Symposium Proceedings*, pp. 887-892, Montreal, Canada, October 1989.

[21] K.W. Ferrara and V.R. Algazi, "Estimation of blood flow velocity using the wideband maximum likelihood estimator," *IEEE 1989 Ultrasonics Symposium Proceedings*, pp. 893-901, Montreal,

Canada, Ocober 1989.

[22] Horn, K.P. and B.G. Schunck, "Determining Optical Flow," *Artificial Intelligence*, vol. 17, pp. 185-203, August 1981.

[23] G.E. Mailloux, F. Langlois, P.Y. Simard, and M. Bertrand, "Restoration of the velocity field of the heart from two dimensional echocardiograms," *IEEE Transactions on Medical Imaging*, vol. 8, no.2, pp. 143-153, June 1989.

[24] Mailloux, G.E., Bleau, A., Bertrand, M., and R. Petitclerc, "Computer analysis of heart motion from two-dimensional echocardiograms," *IEEE Transactions on Biomedical Engineering*, vol. BME-34, no. 5, pp. 356-364, May 1987.

[25] M. Bertrand, J. Meunier, M. Doucet, and G. Ferland, "Ultrasonic biomechanical strain gauge based on speckle tracking," *1989 Ultrasonics Symposium Proceedings*, pp. 859-863, Montreal, Canada, 1989.

[26] W.M. Gardiner, and M.D. Fox, "Color-flow US imaging through the analysis of speckle motion," *Radiology*, vol. 172, pp. 866-868, 1989.

[27] G.E. Trahey, S.M. Hubbard, and O.T. von Romm, "Angle independent ultrasonic blood flow detection by frame-to-frame correlation of B-mode images," *Ultrasonics*, vol. 26, pp. 271-276, September 1988.

[28] L. Bohs and G.E. Trahey, "An efficient technique for two-dimensional ultrasonic velocity imaging", *Ultrasonic Imaging*, vol. 12, no. 2, p. 133, 1990 (abstract).

[29] D. Kim, T.M. Kinter, and J.F. Greenleaf, "Correlation search method with third-order statistics for computing velocities from ultrasound images," *IEEE 1989 Ultrasonics Symposium Proceedings*, pp. 869-872, Montreal, Canada, 1989.

[30] Operating Manual for SWF-5RD Electromagnetic Square Wave Flowmeter, Zepeda Instruments Co., Seattle, WA, 1987.

[31] P.M. Embree,, "UDAS - Ultrasound Data Aquisition System," Internal BRL report, University of Illinois at Urbana-Champaign, Urbana, IL, 1985.

[32] D.A. McDonald, *Blood Flow in Arteries*. London: Edward Arnold Publishers Ltd., 1960.

[33] WAAG II User Guide, Markenrich Corporation, Duarte, CA, 1988.

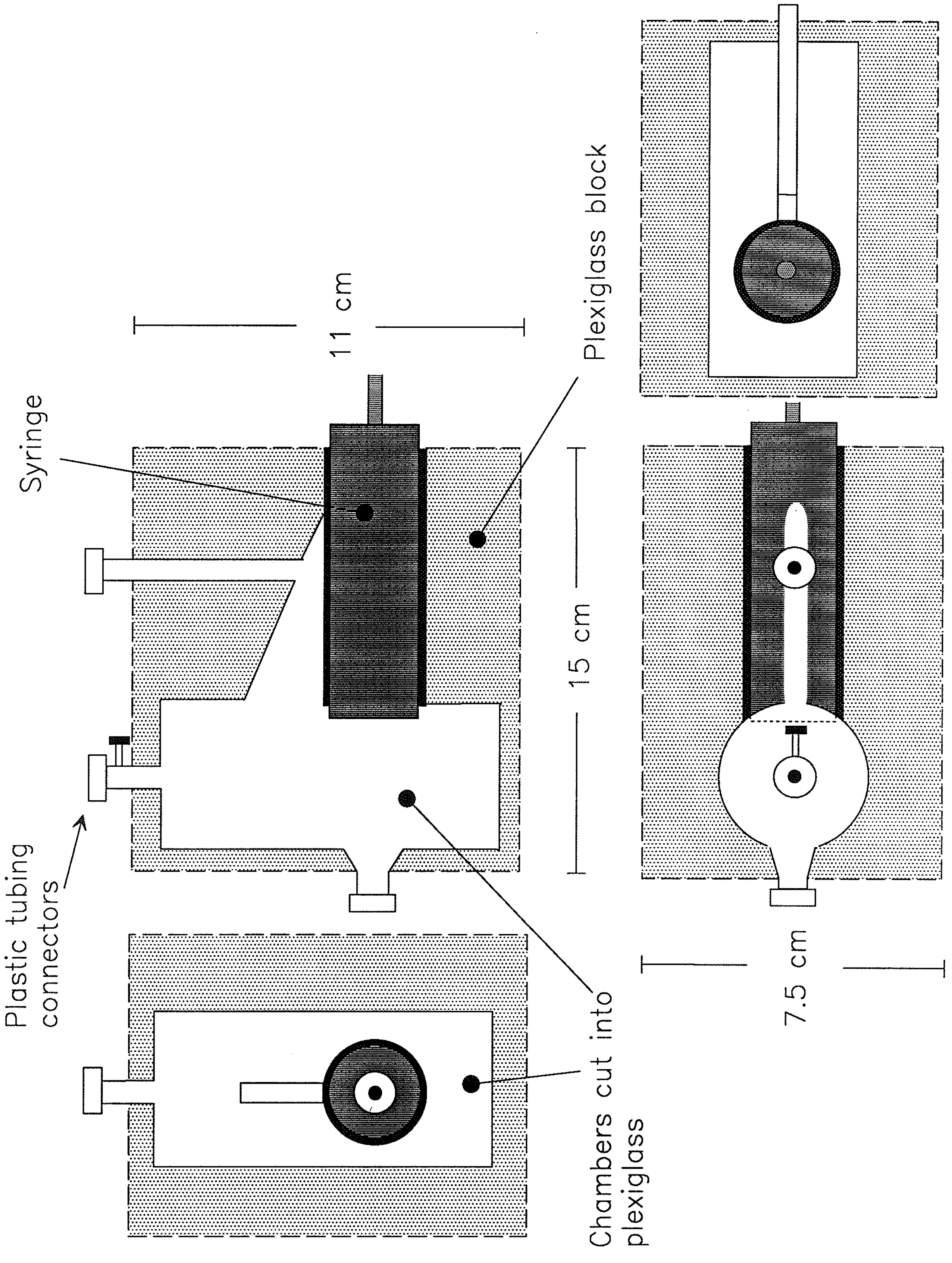
[34] Personal correspondence with J. Zachary, DVM.

[36] J.T. Chen, "Design and implementation of a high-speed residue number system correlator for ultrasonic time domain blood flow measurement," M.S. thesis, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL, 1990.

[37] M.A. Soderstrand, W.K. Jenkins, G.A. Jullien,, and F.J. Taylor, Eds., *Residue Number System Arithmetic, Modern Applications in Digital Signal Processing*. New York: IEEE Press, 1986.

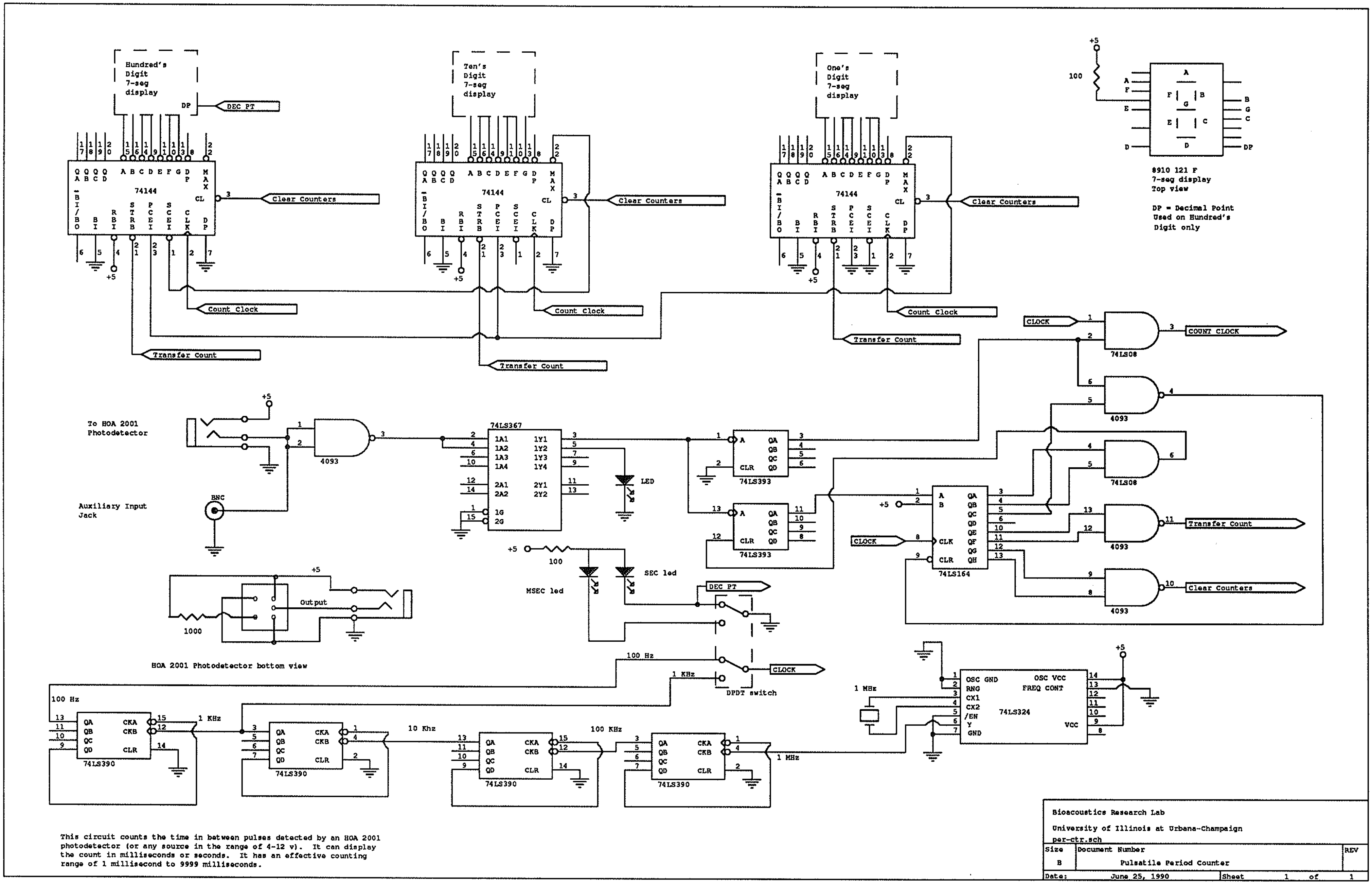
[38] Personal correspondence with Joyce Bender-Schmale, sonographer.

APPENDIX A
DETAILS OF PULSING UNIT



APPENDIX B

PULSATILE PERIOD COUNTER SCHEMATIC



This circuit counts the time in between pulses detected by an HOA 2001 photodetector (or any source in the range of 4-12 v). It can display the count in milliseconds or seconds. It has an effective counting range of 1 millisecond to 9999 milliseconds.

Bioacoustics Research Lab

University of Illinois at Urbana-Champaign

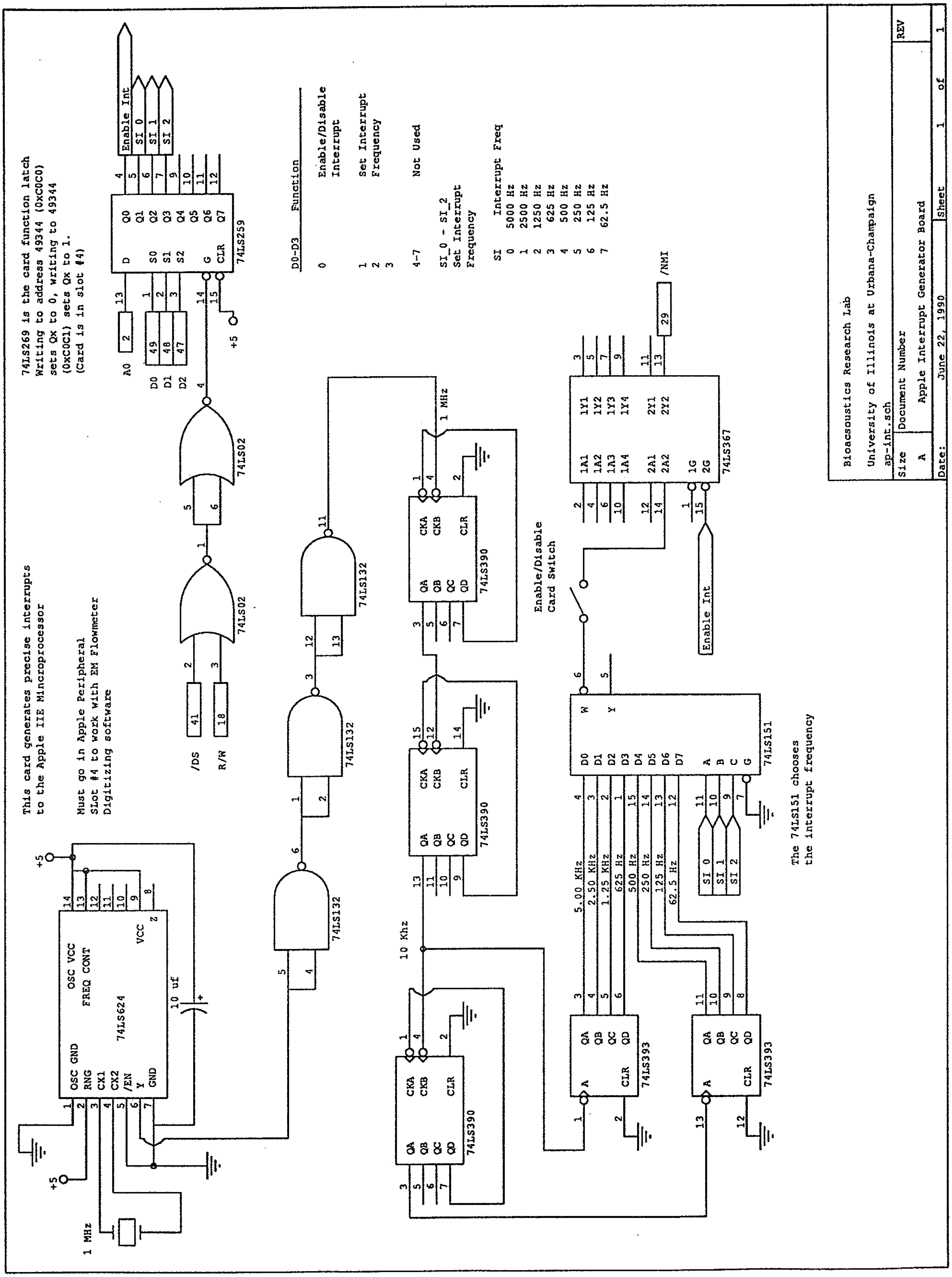
per-ctr.sch

Size	Document Number	REV
B	Pulsatile Period Counter	

Date: June 25, 1990 Sheet 1 of 1

APPENDIX C

APPLE IIE INTERRUPT GENERATOR BOARD



This card generates precise interrupts
to the Apple IIE Mincroprocessor

Must go in Apple Peripheral
SLot #4 to work with EM Flowmeter
Digitizing software

74LS269 is the card function latch
Writing to address 49344 (0xC0C0)
sets Qx to 0, writing to 49344
(0xC0C1) sets Qx to 1.
(Card is in slot #4)

D0-D3	Function
0	Enable/Disable Interrupt
1 2 3	Set Interrupt Frequency
4-7	Not Used

SI_0 - SI_2
Set Interrupt
Frequency

SI	Interrupt Freq
0	5000 Hz
1	2500 Hz
2	1250 Hz
3	625 Hz
4	500 Hz
5	250 Hz
6	125 Hz
7	62.5 Hz

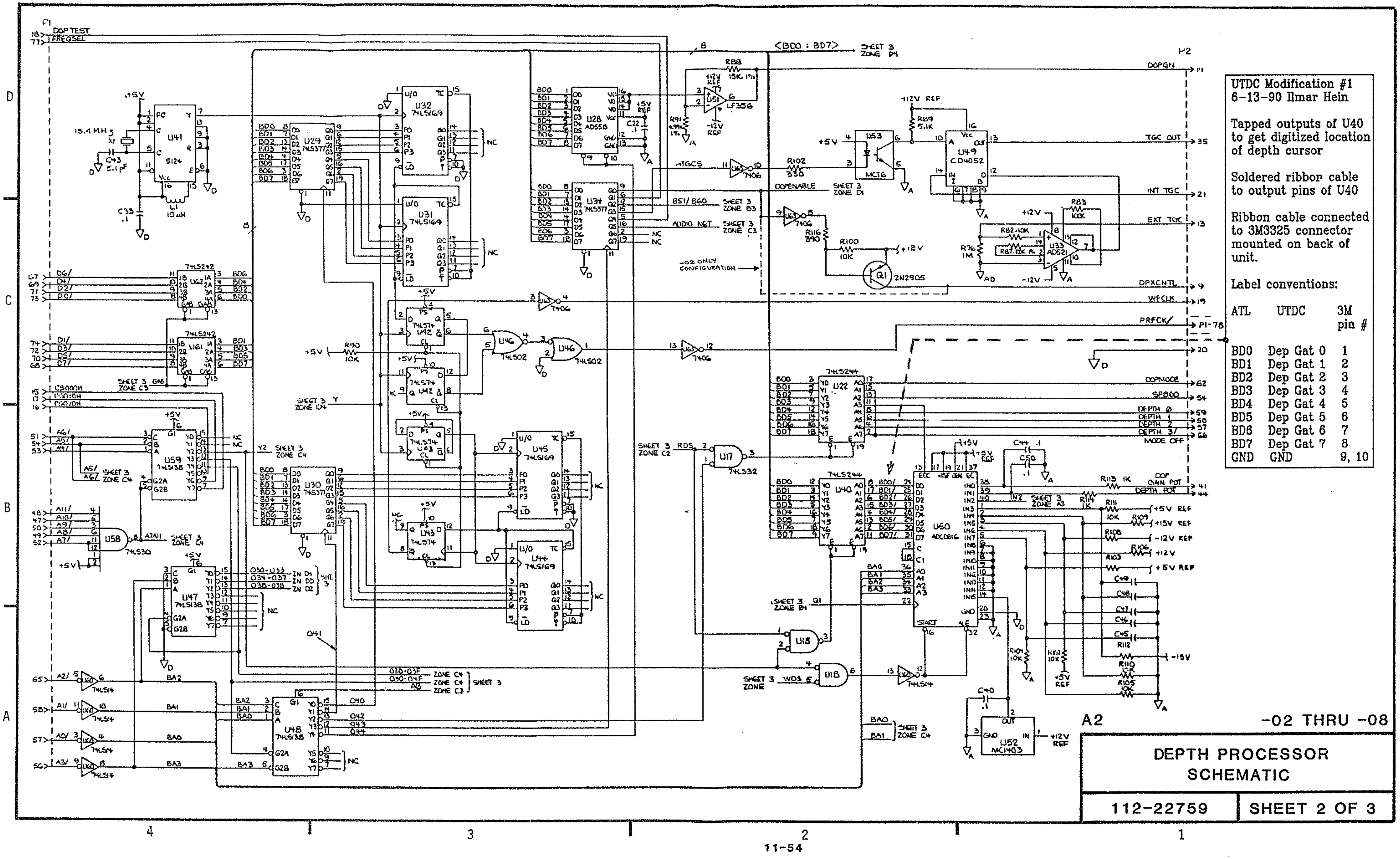
The 74LS151 chooses
the interrupt frequency

Bioacoustics Research Lab
University of Illinois at Urbana-Champaign
ap-int.sch

Size	Document Number	REV
A	Apple Interrupt Generator Board	
Date: June 22, 1990	Sheet 1 of 1	

APPENDIX D

MODIFICATIONS TO ATL MK500 SYSTEM



UTDC Modification #1
6-13-90 Ilmar Hein

Tapped outputs of U40 to get digitized location of depth cursor

Soldered ribbon cable to output pins of U40

Ribbon cable connected to 3M3325 connector mounted on back of unit.

Label conventions:

ATL	UTDC	3M pin #
BD0	Dep Gat 0	1
BD1	Dep Gat 1	2
BD2	Dep Gat 2	3
BD3	Dep Gat 3	4
BD4	Dep Gat 4	5
BD5	Dep Gat 5	6
BD6	Dep Gat 6	7
BD7	Dep Gat 7	8
GND	GND	9, 10

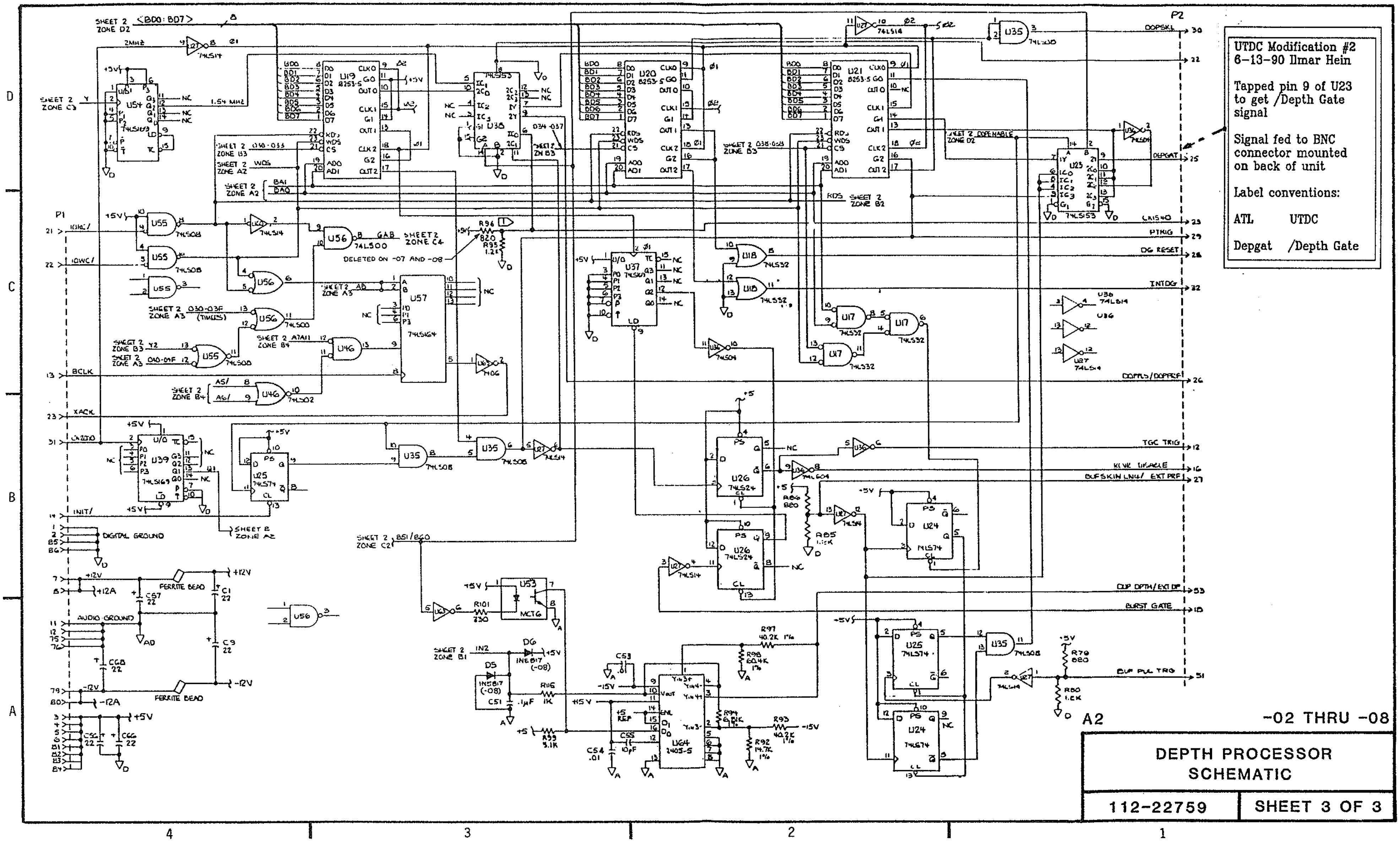
A2

-02 THRU -08

DEPTH PROCESSOR SCHEMATIC

112-22759

SHEET 2 OF 3



UTDC Modification #2
6-13-90 Ilmar Hein

Tapped pin 9 of U23 to get /Depth Gate signal

Signal fed to BNC connector mounted on back of unit

Label conventions:

ATL UTDC

Depgat /Depth Gate

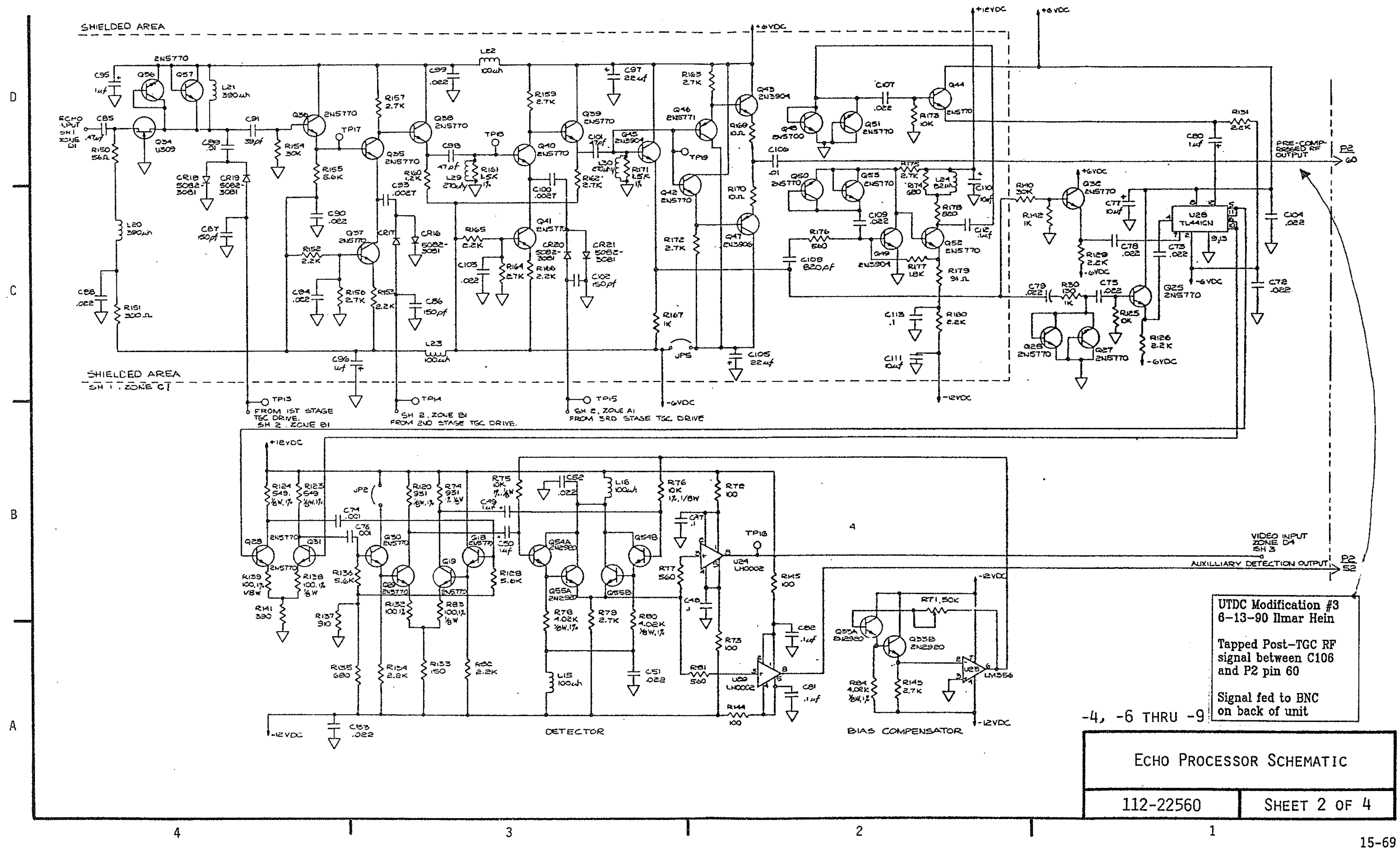
A2

-02 THRU -08

DEPTH PROCESSOR SCHEMATIC

112-22759

SHEET 3 OF 3



UTDC Modification #3
6-13-90 Ilmar Hein

Tapped Post-TGC RF signal between C106 and P2 pin 60

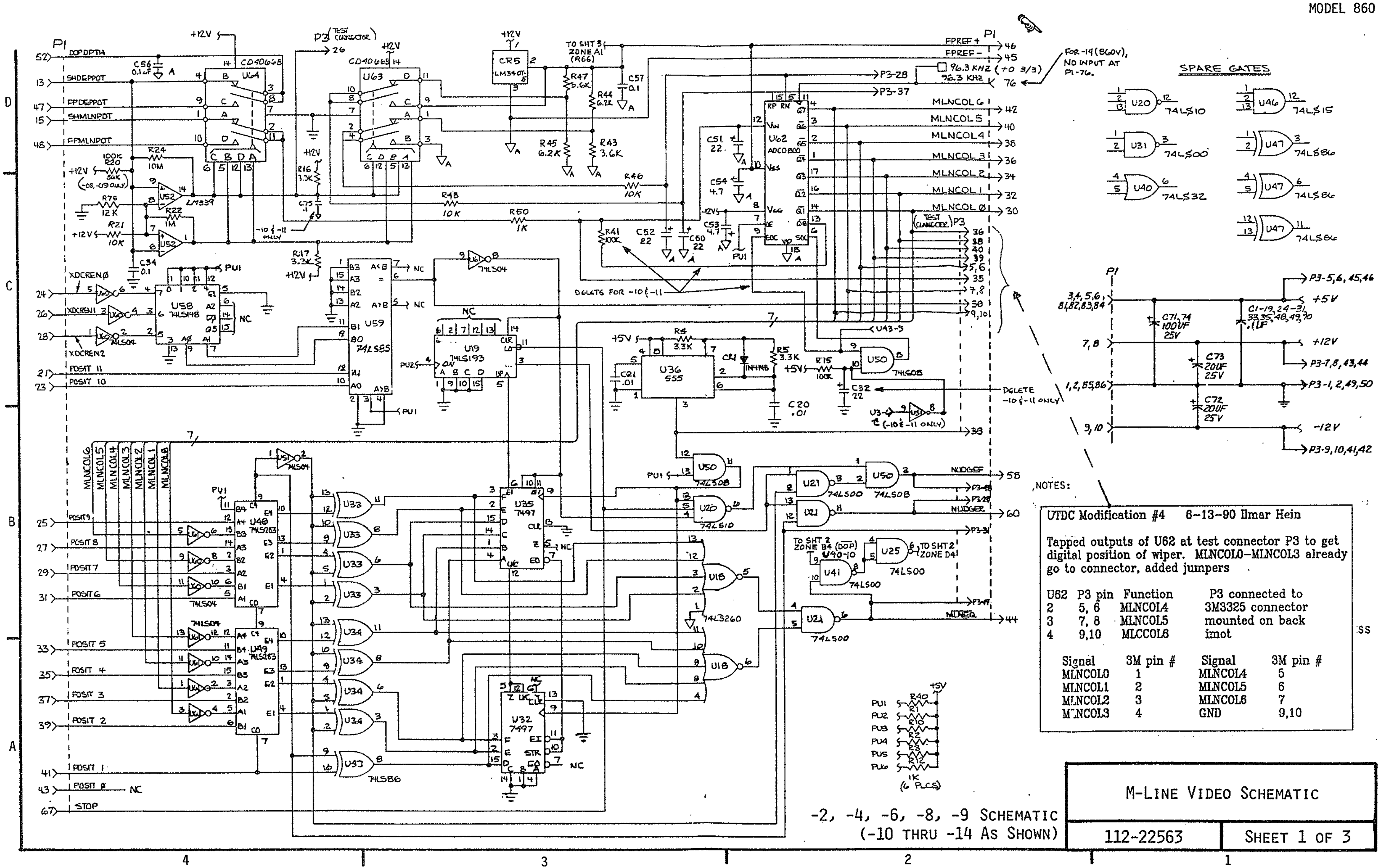
Signal fed to BNC on back of unit

-4, -6 THRU -9

ECHO PROCESSOR SCHEMATIC

112-22560	SHEET 2 OF 4

15-69



NOTES:

UTDC Modification #4   6-13-90 Ilmar Hein

Tapped outputs of U62 at test connector P3 to get digital position of wiper. MLNCOL0-MLNCOL3 already go to connector, added jumpers

U62	P3 pin	Function	P3 connected to
2	5, 6	MLNCOL4	3M3325 connector
3	7, 8	MLNCOL5	mounted on back
4	9,10	MLCCOL6	imot

Signal	3M pin #	Signal	3M pin #
MLNCOL0	1	MLNCOL4	5
MLNCOL1	2	MLNCOL5	6
MLNCOL2	3	MLNCOL6	7
M'NCOL3	4	GND	9,10

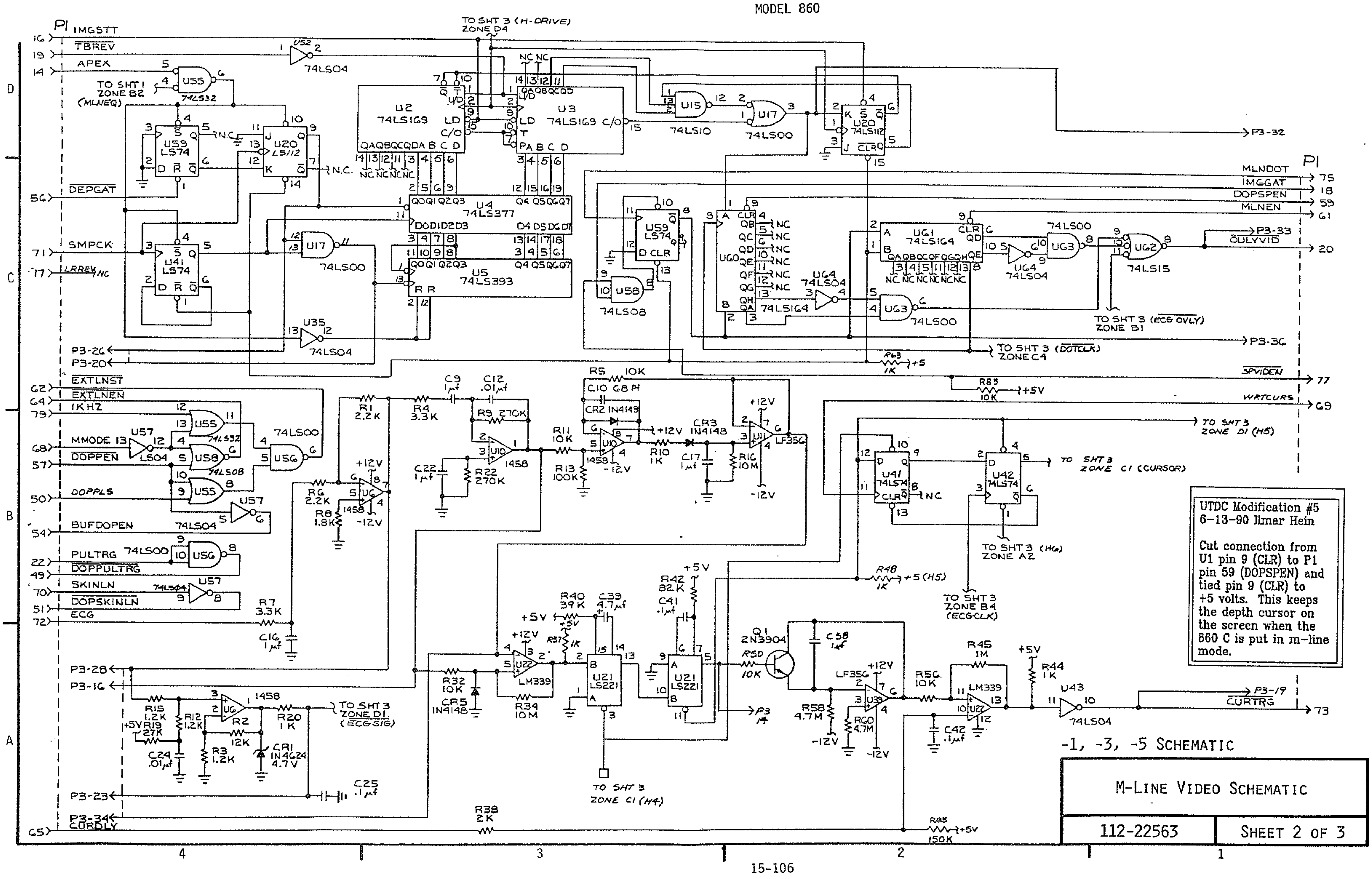
-2, -4, -6, -8, -9 SCHEMATIC
(-10 THRU -14 AS SHOWN)

M-LINE VIDEO SCHEMATIC

112-22563   SHEET 1 OF 3

Change 3   15-101

MODEL 860



UTDC Modification #5
6-13-90 Ilmar Hein

Cut connection from U1 pin 9 (CLR) to P1 pin 59 (DOPSPEN) and tied pin 9 (CLR) to +5 volts. This keeps the depth cursor on the screen when the B60 C is put in m-line mode.

-1, -3, -5 SCHEMATIC

M-LINE VIDEO SCHEMATIC

112-22563

SHEET 2 OF 3

APPENDIX E

TEMPORARY SYSTEM PROGRAMS

```
rem     ***** Temporary system UTDC flowmeter DOS batch file *****

rem     This batch file calls four programs which perform different
rem     blood flowmeter functions

rem     ACQUIRE acquires the echoes  (Microsoft C)
rem     CALCVAR does the correlations and calculates velocities (HighC 386)
rem     PLOTVEL plots the velocity vs range (Microsoft C & GraphiC 4.1)
rem     STORE   renames datafile names MK500.ECH and MK500.PLT to the names
rem             xx.ECH and xx.PLT where xx is the number located in FILENUM.MK5
rem             and also increments value in FILENUM.MK5


rem     set paths:

SET GPC=\C\GRAPHIC\FONTS
path=\hc;\dos;\pe2;\hc\inc;\bat;\nort;\mk500
set ipath=\hc\inc\
cd \mk500\var


rem     main blood flowmeter loop:

:loop
      acquire
      run386 calcvar
      plotvel
      store
goto loop
```

```
/******************************************* AQUIRE3.C ****************************************/
/*                                                                                            */
/*   This program is used with the 40 MHz WAAG II A/D board.  It will  continuously sample    */
/*   and display RF echoes until a key is pressed.  Once  a key is pressed, it will acquire    */
/*   20 1K RF echoes and store them to disk.  The name of the file the data is stored in is   */
/*   called MK500.ECH.                                                                        */
/*                                                                                            */
/*   The data is stored as a single signed integer in the range of -127 to +128.              */
/*                                                                                            */
/*   The data aquisition is triggered from the /Depth_Gate pulsefrom the MK500 system.        */
/*                                                                                            */
/*   The MK500 wiper and depth cursor locations are read in as 8-bit numbers using the        */
/*   METRABYTE PDMA-16 card.                                                                  */
/*                                                                                            */
/**********************************************************************************************/


#include <graphms.h>
#include <dos.h>
#include <malloc.h>
#include <stdlib.h>
#include <stdio.h>


/* WAAG2 locations */

#define    WAAG     0xD0000000  /* WAAG memory */
#define    PORT0    0x178       /* default setting      */
#define    PORT1    0x179       /* all switches are off */
#define    PORT2    0x17A
#define    PORT3    0x17B

#define LENGTH 1024       /* Echo Length */
#define COUNT -1279       /* Value to load WAAG counter is -(LENGTH + 0xFF ) */
#define CONTROL 0xA00C    /* Set aquisition at 40 Mhz */


/* PDMA 16 card addresses */

#define PDMA_A 0x300         /* address of port a */
#define PDMA_B 0x301         /* address of bort b */

#define PDMA_DMA 0x302       /* DMA control register */
#define PDMA_INT 0x303       /* Interrupt control register */

int
    control = 0xA00c         /* control parameters for WAAG board.  */
    ;                        /* set for 40 MHz operation            */


char
     trigger_level = 0x80    /* trigger level (not used since triggering from MK500) */
     ;

char huge *echo;            /* Array where echo data is to be stored */

/*********************************** main ************************************************/
/*                                                                                        */
/*  This main routine is used only to allocate memory for the echo data.  Enough  memory is */
/*  allocated for 128 1K echoes, though only 20 echoes are ever actually acquired.         */
/*                                                                                        */
/******************************************************************************************/
main()
{
 echo = (unsigned char huge *) halloc( 131072, sizeof(char) );
  if(echo == NULL ) { printf("\n unable to halloc echoes \n"); exit (0); }
  printf("\n Memory left = %u \n", _memavl() );
 outp( PDMA_DMA, 0 );
 outp( PDMA_INT, 0 );
 go( echo );
}
```

```
/******************************************** go ********************************************/
/*                                                                                            */
/*  This is really the main program.                                                          */
/*                                                                                            */
/**********************************************************************************************/


go( echoes )

 char huge
                  echoes [] [1024]   /* array where echoes are stored */
                  ;
{
     unsigned char
                  *waag,             /* pointer to WAAG's memory   */
                  erase[700],        /* buffer used to erase trace */
                  depth,             /* cursor depth variable      */
                  angle              /* cursor angle variable      */
                  ;
     char
           dummy,
           key                       /* used for checking keybpresses */
           ;

     int
           c,
           i,
           size=330,                    /* number of sample points */
           file_number,                 /* current datafile number */
           total_echoes = 20,           /* echoes to acquire       */
           trig,
           trigger                      /* WAAG memory address where trigger */
           ;                            /* has occured (from /Depth_Gate)    */


     void
          load_trigger_offset()         /* routine loads trigger offset (not used since triggered from MK500) */
          ;

     int
         sample()                       /* routine that acquires a 1K echo */
         ;

     FILE *fptr;


     for(c=0; c<700; c++) erase[c] = 1;         /* initialize erase trace buffer */

     waag = (char *) WAAG;                      /* pointer to WAAG memory */

     _setvideomode(_VRES16COLOR);               /* Set video mode to VGA       */
     _clearscreen( _GCLEARSCREEN );             /* Draw signal clipping limits */
     _setcolor(9);                              /* and menu                    */
     _moveto(0,99);
     _lineto(600,99);
     _moveto(0, 100+256);
     _lineto(600,100+256);
     _settextcolor(1);
     _settextposition(24,1);


     if( (fptr = fopen( "filenum.mk5", "r" )) == NULL )      /* file where current datafile # is stored */
       {
        printf("\n Can't fine filenum.mk5!\n");
        exit(0);
       }

     fscanf(fptr, "%d", &file_number);
     fclose(fptr);
     printf(" 40 MHz RF Aquisition          ***  %d  ***\n", file_number );
```

```
    /*------------------------------ plot RF echo loop -----------------------------------*/
    /*                                                                                     */
    /* This while loop continuously plots echoes until the keyboard is hit. When that hap-  */
    /* pens, this loop is exited and data is acquired and stored to disk.                  */
    /*                                                                                     */
    /* --------------------------------------------------------------------------------*/

     while( !kbhit() )                            /* loop until keyboard hit */

       {
        trigger = sample();                       /* data acquisition     */
        trigger <<= 1;                            /* address = trigger*2  */
        angle = inp (PDMA_B);                     /* get wiper angle      */
        depth = inp(PDMA_A);                      /* get depth of cursor  */

        _settextposition(1,1);
        printf("Angle = %d     ",angle);
        printf("Depth = %d  ", depth);

        c=0;                                      /* Erase previous trace */
        _setcolor(0);
        _moveto( c,(int) erase[0]+100 );
        for (c=1; c<600; c++)
            {
             _lineto( c, (int)erase[c]+100 );
            }

        _setcolor(13);                                      /* plot trace */
        c=0;
        _moveto( c, (int)*( waag + c + trigger ) + 100 );      /* move to 1st pt */

        for (c=1; c<600; c++)
            {
             _lineto( c, (int)*( waag + c + trigger ) + 100 );
            }

        memcpy(erase, (waag+trigger), 600 );                 /* copy trace to erase buffer */
       }  /* main while loop */

       getch();                                             /* clear char if kbard hit */

       /-------------------------------- acquire and send data to a file -----------------------------*/

       printf("\n Aquiring Data...\n");

       aquire( total_echoes, echoes, angle, depth );

       printf("\nDone Aquiring!\n");
       printf("Writing to disk...");

       if( ( fptr = fopen("mk500.ech","w") ) == NULL )     /* The data is stored in the file MK500.ECH.  A*/
         {                                                  /* separate program is used to read FILENUM.MK5 */
          printf("\n Unable to open echo file for writing...");
          exit(0);
         }
       fprintf(fptr,"%d\n", total_echoes);
       fprintf(fptr,"%d %d\n", angle, depth );
       for( c=0; c< total_echoes; c++ )
          {
           fprintf(fptr,"\n");
           for( i=0; i<1024; i++ )
              {
               dummy = echoes [c] [i] -128;            /* data from WAAG memory is not in 2's complement */
                  fprintf( fptr,"%d\n", dummy );       /* form.  This makes it into 2's complement.      */
              }
          }
          fclose( fptr );
      _setvideomode( _DEFAULTMODE );
      _clearscreen( _GCLEARSCREEN );
}
/*********************** END GO **********************************************************************/
```

```
/************************************ sample ************************************************/
/*                                                                                          */
/*   This routine acquires a 1K echo into the WAAG board's memory.  The location of the     */
/*   start of the 1K echo.  The value returned is 1/2 the trigger address.                  */
/*                                                                                          */
/********************************************************************************************/

 int  sample()
  {
    int
       busy = 1,
       count
       ;

    char
        key
        ;

    outpw(PORT2,0xA06f);               /* initialize */
    outpw(PORT0,0xffff);               /* clear counter */
    outpw(PORT0,0xffff);
    outpw(PORT0,COUNT);                /* load count */
    outpw(PORT0,COUNT);
    outpw(PORT2,CONTROL & 0xbffc);     /* start the counter */

    while (busy)                       /* wait for sampling to be completed */
      {
       if(inp(PORT2) & 01) busy = 0;  /* sampling is done */
      }

    outpw(PORT2,0xA06f);              /* enable  WAAGII RAM   */
    return(inpw(PORT0));              /* read trigger address */
  }


/********************************** END SAMPLE *******************************************/

/******************** LOAD TRIGGER OFFSET ***********************************************/

void  load_trigger_offset(x)
      char x;
  {
     char *waag;
     waag = ( char *) WAAG;

     outpw(PORT2,0xc06f);      /* enabled trigger offset */
     *waag = x;

     outpw(PORT2,0xA06f);
  }

/***************** END LOAD TRIGGER OFFSET *******************************************/

/************************************ acquire ***********************************************/
/*                                                                                          */
/*   This routine acquires 20 echoes and stores them into the array echoes.   It also      */
/*   reads the cursor angle and depth.  The parameters passed are the total number         */
/*   of echoes to be required, and it returns the echo data, cursor angle and depth.       */
/*                                                                                          */
/********************************************************************************************/

aquire( total_echoes, echoes, angle, depth )

  int
      total_echoes
      ;

 unsigned char  huge
                    echoes [ ] [ LENGTH ]
                    ;
  unsigned char
                angle,
                depth
                ;

{
     unsigned char
                   *waag
                   ;
     char
           key
           ;

     int
           c,
           echo_ctr,
           i,
           trigger
           ;

     int
         sample()
         ;


     waag = (char *) WAAG;


     angle = inp( PDMA_B ) & 0x7F;
     depth = inp( PDMA_A );

     for( echo_ctr=0; echo_ctr < total_echoes; echo_ctr++ )
       {

        trigger = sample( );                                        /* acquire a 1K echo     */
        trigger <<= 1;                                              /* address = trigger*2  */

        memcpy( &echoes [echo_ctr] [0], ( waag+trigger ), LENGTH ); /* copy from WAAG memory to echo array */

       }  /* main loop */

}

/********************************** END AQUIRE **********************************************/


/********************** END AQUIRE3.C **********************************************************/
```

```
/******************************** CALCVAR *******************************************************************/
/*                                                                                                          */
/*   This program is a software version of the correlation algorithm using the weighted averaging         */
/*   technique.  It reads in 20 1K echoes, from the file MK500.ech (generated by the AQUIRE program)      */
/*   and calculates velocities from delta spacings of 1 to 10.  The output velocities are written to      */
/*   the data file MK500.VAR. The first line of MK500.var is 1 (only one graph), the second line is 23    */
/*   (23 points in a velocity vs range graph). The rest of the lines have two numbers, the range in mm    */
/*   and the un-angle corrected velocity in cm/s.                                                           */
/*                                                                                                          */
/*   Compiled with Metaware's HighC 386                                                                     */
/*                                                                                                          */
/************************************************************************************************************/


pragma On(Floating_point);
pragma On(387);


#include "stdio.h"
#include "math.h"
#include "stdlib.h"

#include "string.h"

#define DELTA_MAX 10
#define COR_LENGTH 40
#define MIN_SHIFT 2
#define MAX_SHIFT 7

extern float correlate();
extern void corr_20();
extern float calc_sum_squared();



/************************************ tshift ************************************************/
/*                                                                                          */
/*   This routine calculates the time shifts for all deltas and performs the weighted       */
/*   average to get the final velocity result.                                              */
/*                                                                                          */
/********************************************************************************************/

void calc_time_shifts( echo )

int echo[] [1024];

{

int
    angle,           /* Angle measurement was made at               */
    base_echo,       /* base echo address correlations are done from */
    b,               /* counters                                    */
    c,
    d,
    depth,          /* cursor depth at measurement                  */
    range,          /* range correlations are being performed at    */
    test_shift,
    total_echoes    /* total number of echoes (Only 20 are used)    */
    ;


float
      den,             /* array containing flow values                    */
      flow [25],       /* final velocity [range] array                    */
      mean,
      shifts [25][12], /* Array for shifts [range][delta]                 */
      shvar[11],       /* variance of a given delta shift                 */
      tshift [11],
      tshift_2 [25]
      ;
```

```
FILE *fptr;

printf("\nReading from disk...\n");
if (( fptr = fopen("mk500.ech","r"))==NULL) { printf("\n error \n"); exit(1); }

   fscanf(fptr, "%d", &total_echoes );
   fscanf(fptr, "%d %d", &angle, &depth );
   total_echoes=20;
   printf("\n Echoes = %d  Angle = %d   Depth = %d",total_echoes, angle, depth);


   /*---------------- read in echoes ----------------*/
   for (b=0; b < total_echoes; b++)
       {
        for(c=0; c<1024; c++)
           {
            fscanf( fptr,"%d",&echo[b][c] );
           }
       }
       fclose(fptr);


   if (( fptr = fopen("mk500.var","w"))==NULL) { printf("\n error \n");
   exit(1); }

   base_echo = 0;

   printf("\nStarting correlations...\n");

   b = 0; /* range counter */

/*=================== range loop ===================*/
   for(range = 10; range<=900; range+=40 )
    {

    corr_20( base_echo, echo, range, tshift, tshift_2 );


    /*-------------- calculate mean shift and shift squared values for each delta -----------*/
    for(c=1; c<=10; c++ )
       {
        tshift   [c] = tshift [c] / (float)c;       /* average shift value for a given delta */
        tshift_2 [c] = tshift_2 [c] / (float)c;     /* shift squared value for a given delta */
       }


    /* ------------- calculate variances of shift estimates at each delta -------------------*/
    for(c=1; c<=10; c++)
       {
        shvar [c] = tshift_2 [c] - tshift[c] * tshift [c];
        if( shvar [c] <= 0.0 ) shvar [c] = 10.0 * fabs ( tshift [c] );
        if( fabs( tshift [c] ) < .01 ) tshift [c] = .01;
       }

   mean = 0.0;
   den = 0.0;

   /*---------------------- calculate weighted average over all 10 deltas ---------------------*/
   for( c=1; c<=10; c++ )
      {
       den  += 1.0/( shvar [c] );
       mean += tshift [c] / shvar [c];
      }
   flow [b] = mean / den;


   b++;
  }
  /*====================== end range loop ======================*/
  /*------------ write data to disk ---------------------*/
  printf("\nWriting correlation results to disk....\n");

  fprintf(fptr,"1\n");           /* Total plots     */
  fprintf(fptr,"23\n");          /* Points per plot  */

  fprintf(fptr,"\n");

  for( d=0; d <b; d++)       /* range loop      */
     {
      fprintf( fptr, "%f %f\n", (float)(d*40+10)*.02, flow [d] *1.414); /* assume 45 deg, prf = 500           */
     }                                                                  /* 1.414 = c / ( 2* PRP * cos(45))    */
                                                                        /* range (cm) = (d*40+10)*25nsec*c /2 */

   fclose( fptr );
   printf("\nDONE!\n");
}
/************************************* end tshift ***************************************************************/

/***************************** corr_20 ***********************************************/
/*                                                                                   */
/*      Correlates a block of 20 echoes, from delta = 1 to 10                        */
/*                                                                                   */
/*************************************************************************************/


void corr_20 ( base_echo, echo, range, tshift, tshift_2 )

int
    base_echo,
    echo [] [1024],
    range
    ;

float  tshift  [],             /* time shift        */
       tshift_2 []             /* time shift squared */
       ;

{

   int
       c,
       cor_ctr,                      /* correlation counter. Used in searching for maximum correlation */
       delta,                        /* current delta value                                            */
       offset,                       /* current offset value                                           */
       shift,                        /* current shift value                                            */
       start_index [ DELTA_MAX +1 ]  /* starting location within echo to start correlations
       ;


   float
         correlation [10],               /* values of correlations as searching for max  */
         guess,                          /* starting guess of correlation                */
         peak,                           /* peak of parabolic fit to 3 pts               */
         peak_sum   [ DELTA_MAX +1  ],   /* sum of peak values for a given delta         */
         peak_sum_2 [ DELTA_MAX + 1 ],   /* sum of peak squared values for a given delta */
         sum_e1_x_e2,                    /* sum of 1st echo * 2nd echo over COR_LENGTH   */
         sum2_e1,                        /* sum of squares of 1st echo over COR_LENGTH   */
         sum2_e2                         /* sum of squares of 2nd echo over COR_LENGTH   */
         ;

    for( c=0; c<=DELTA_MAX; c++ )       /* initialize variables */
       {
        peak_sum [c] = 0;
        start_index [c] = 0;
        tshift   [c] = 0;
        tshift_2 [c] = 0;
       }
    for( delta=1; delta <= DELTA_MAX; delta++ )
     {

      guess =  tshift [ delta - 1 ] * ( ( (float)delta+1.0 ) / (float)delta );

      for( offset=0; offset < DELTA_MAX; offset++ )
       {
        shift = (int)( guess ) - MIN_SHIFT;

        cor_ctr = 0;
        sum2_e1 = calc_sum_squared( &echo [ base_echo+offset ] [ start_index [offset] + range ] );


while( (
         sum_e1_x_e2 = correlate( &echo [ base_echo+offset ]       [ start_index [offset] + range ],
                                  &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] )
         ) < 0.0 && cor_ctr < 3 )
         {
          shift++;
          cor_ctr++;

         }

     if( cor_ctr < 6 )    /* Give up if 3 neagtive cors */
        {
         sum2_e2 = calc_sum_squared( &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] );

         correlation [cor_ctr] = sum_e1_x_e2 / (float)sqrt( (double)(sum2_e1 * sum2_e2) );      /* prev2 */

         shift++;
         cor_ctr++;

         sum_e1_x_e2 = correlate( &echo [ base_echo+offset ]       [ start_index [offset] + range ],
                                  &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] );

         sum2_e2 = calc_sum_squared( &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] );

         correlation [cor_ctr] = sum_e1_x_e2 / (float)sqrt( (double)(sum2_e1 * sum2_e2 )); /* prev1 */

        while( cor_ctr < MAX_SHIFT )
         {
          shift++;
          cor_ctr++;

          sum_e1_x_e2 = correlate( &echo [ base_echo+offset ]       [ start_index [offset] + range ],
                                  &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] );

          sum2_e2 = calc_sum_squared( &echo [ base_echo+offset+delta ] [ start_index [offset] + shift + range ] );

          correlation [cor_ctr] = sum_e1_x_e2 / (float)sqrt( (float)(sum2_e1 * sum2_e2) );

          if( correlation [cor_ctr] < correlation [cor_ctr - 1] ) break;
         }

         if(correlation [cor_ctr] < 0 ) peak = shift -1;
         else if( correlation [cor_ctr-1] < correlation [cor_ctr-2] ) peak = shift-2;
         else
         peak =  (float)shift-1 + ( correlation [ cor_ctr-2 ] - correlation [ cor_ctr ] )
                                /( 2*( correlation [cor_ctr-2]-2*correlation [cor_ctr-1] + correlation [cor_ctr]));

         if( peak < -20.0 || peak > 63.0 ) peak = shift - 1;
        }
        else peak = shift - 4 ;     /* if  neg cors, give up and set shift = 0 */

        peak_sum   [ delta ] += peak;
        peak_sum_2 [ delta ] += peak*peak;

        if( delta == 1 ) start_index [ offset+1 ] = peak_sum [delta];

       }    /* offset loop */
```

```
        tshift   [delta] = peak_sum   [delta] / DELTA_MAX;
        tshift_2 [delta] = peak_sum_2 [delta] / DELTA_MAX;

      }       /* delta loop */
}
/********************* end corr20 *****************************************/
/******************************* correlate **********************/
/*                                                               */
/* Correlates two echo sections x and y.  Length is 40 pts long  */
/* Multiplies each element of array x with the corresponding     */
/* element of y and adds all the corresponding multiplications.  */
/*                                                               */
/****************************************************************/

float correlate ( x, y )

int x [], y[];

{

float sum_xy
      ;

  int c;


  for( c=0; c < COR_LENGTH; c++ )
     {
      sum_xy += (float)x [c] * (float)y [c] ;
     }

  return( sum_xy );

}
/************************ end correlate ***********************************/


/******************************* calc_sum_squared *******************/
/*                                                                  */
/*       Performs the sum of squares over the range of an echo      */
/*                                                                  */
/********************************************************************/

float calc_sum_squared( echo )

int echo [ COR_LENGTH ];

{
  int
      a
      ;


  float
        sum2 = 0;
        ;

  for( a=0; a < COR_LENGTH; a++ )
     {
      sum2 += echo[a] * echo[a];
     }

  return( sum2 );

}
/************************** end clac_sum_squared **************************/
```

```
/********************************** main ************************************/
/*                                                                          */
/*  The real purpose of main is to allocate memory for 256 1K echoes       */
/*  Only 20 1K echoes are really used                                       */
/*                                                                          */
/****************************************************************************/

void main ()

{

  int *echoes;

  echoes =  malloc( 262144 );

  if(echoes == NULL) { printf("\n Unable to malloc echoes \n"); exit (1); }
  calc_time_shifts( echoes );

}

/*************************************** end calcvar.c ****************************************/
^Z
```

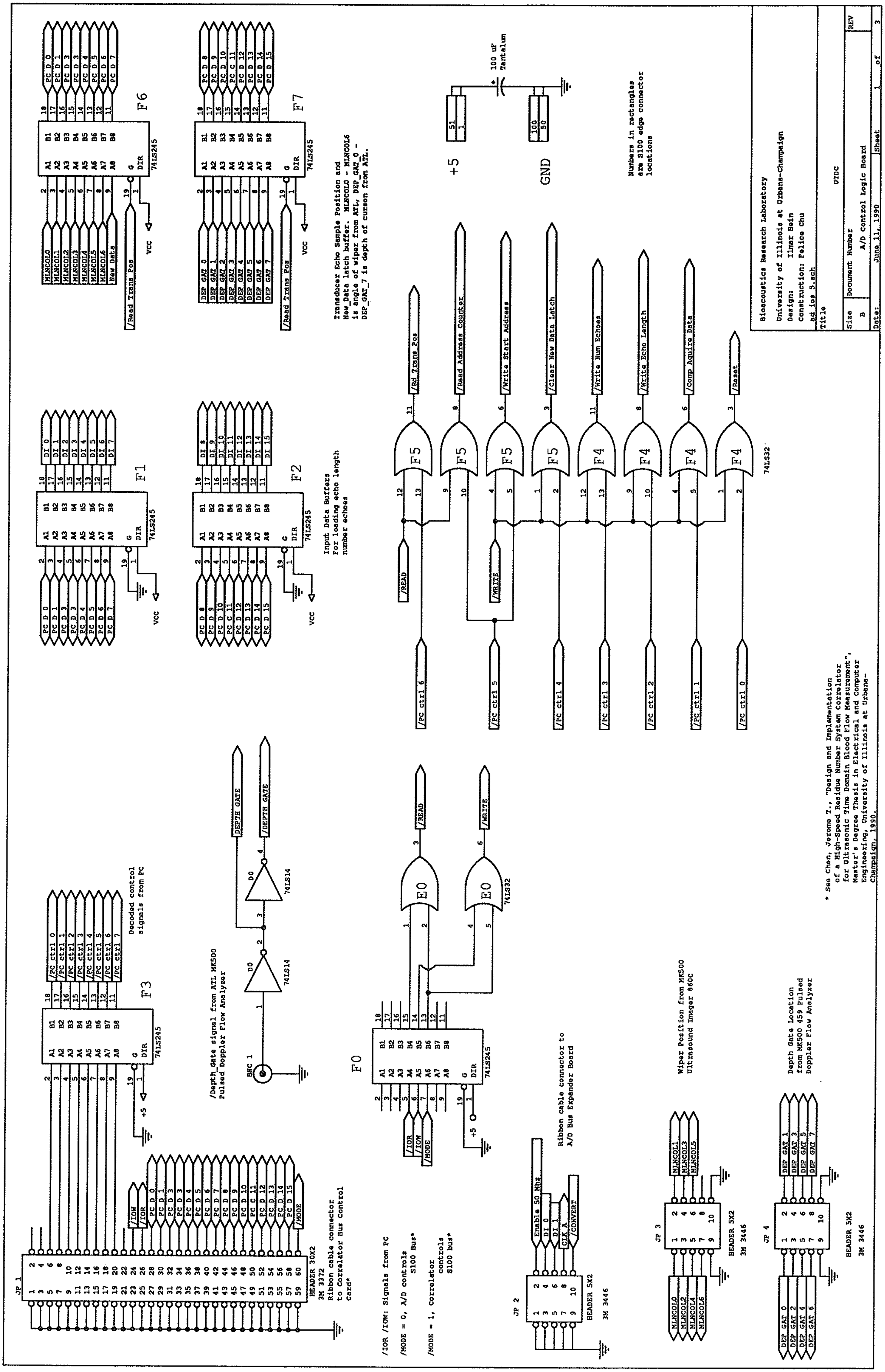
```
/*********************************************** PLOTVEL.C ***********************************************/
/*                                                                                                          */
/*    This program reads in the data file mk500.var (generated by the CALCVAR program).                     */
/*    Generates output plot file MK500.plt                                                                  */
/*                                                                                                          */
/*    Adapted from a generic program which can read in data for N graphs of M points long.                  */
/*    Uses routines from GraphiC 4.1, Scientific Endeavors Corp.                                            */
/*                                                                                                          */
/************************************************************************************************************/

#include "graph.h"
#include "stdio.h"
#include "stdlib.h"

main(argc, argv)
    int argc;
    char *argv[];

{
      float x[100], y[100],                  /* plot buffers                                            */
            xi[10] [100], yi[10] [100],      /* input data arrays. can plot 10 graphs with 100 pts each. */
            xmax,xmin, ymax, ymin            /* max and min data values used in scaling axes            */
            ;

      int c,              /* counter variables */
          i,
          j,
          sym=1,          /* plot symbol type              */
          fntc,           /* font number                   */
          nxdiv,          /* number of x divisions of graph  */
          nydiv,          /* number of y divisions of graph  */
          npts,           /* number of points in plot      */
          numplots,       /* numper of graphs to be plotted  */
          numpoints [20]  /* number of data points per graph */
          ;


      char fnts,        /* font type            */
           fnme[15],    /* data file name       */
           xnme[30],    /* x-axis name          */
           ynme[30],    /* y-axis name          */
           title[30],   /* title of plot        */
           pltfl[15]    /* output plot file name */
           ;

      FILE *fptr;

      strcpy(fnme,"mk500.var");
      strcpy(pltfl,"mk500.plt");

      if( (fptr=fopen(fnme,"r"))==NULL)
        { printf("Can't find %s file!\n",fnme);  exit(); }

      strcpy(fnt,"\312"); fnts = '\312';  /* choose font */

      strcpy(title,fnt);
      strcpy(xnme,fnt);
      strcpy(ynme,fnt);

      strcpy(title," ");
      strcpy(xnme,"Range (mm)");
      strcpy(ynme,"Velocity (cm/sec)");

      /*------------ read in number of plots and points per plot -----------*/
      fscanf(fptr,"%d",&numplots);
      for(j=0; j<numplots; j++)
          fscanf(fptr,"%d",&numpoints[j]);
    /*-------------------- read in velocity data --------------------------*/
    /* also find maximum and minimum values */
    xmax=0;
    ymax=0;
```

```
      for(i=0; i<numplots; i++)
         {
           for(j=0; j<numpoints[i]; j++)
            {
             fscanf(fptr,"%g %g",&xi[i] [j], &yi[i] [j]);
             if(i==0 && j==0){ymax=yi[i][j];xmax=xi[i][j];xmin=xmax;}
             if(xi[i][j]>xmax)xmax=xi[i][j];
             if(yi[i][j]>ymax)ymax=yi[i][j];
             if(xi[i][j]<xmin)xmin=xi[i][j];
             }
         }

         ymin = .3;                            /* scales axes better */

       /*---------------------- start plotting routines ----------------------*/
         bgnplot(1,'g',pltfl);
         startplot(8);
         font(4,"microg.fnt",'\310',"triplex.fnt",'\311',"complex.fnt",'\312'
                 ,"duplex.fnt",'\313');


         page(9.0,6.855);    /* Plot page sizes */
         area2d(8.0,5.5);

         grid(1);             /* plot a grid  */
         upright(1);          /* upright axes */
         nxdiv=10;            /* number of x divisions */
         nydiv=10;            /* number of y divisions */
         color(9);

         /*------------------- copy data into plot buffer y[ ] ------------------------*/
         npts=numpoints[0];
         for (j=0; j<npts; j++){ x[j]=xi[0] [j]; y[j]=yi[0] [j];}
         y[npts]=ymax; x[npts]=xmax;
         y[npts+1]=ymin;x[npts+1]=xmin;

         scales(nxdiv,nydiv,x,y,npts+2);                      /* scale the axes        */

         heading(title);                                      /* headings & axis labels */
         yname(ynme);
         xname(xnme);

         /*----------------- plot the data ---------------------------------------------*/
         for (i=0; i<numplots; i++)
             {
              if(i+9 <= 15) c = i+9;                           /* set the color */
              else c = i+9-7;
              c=14;
              color(c);

              sympick(i);                                       /* choose symbol */
              npts=numpoints[i];
              for(j=0; j<npts; j++)
               {  x[j]=xi[i] [j]; y[j]=yi[i] [j]; }

              curve(x,y,npts,sym);                              /* plot the points */
             }

        endplot();       /* exit plotting routines */
        stopplot();
}

/************************************* END PLOTVEL.C ***********************************/
```

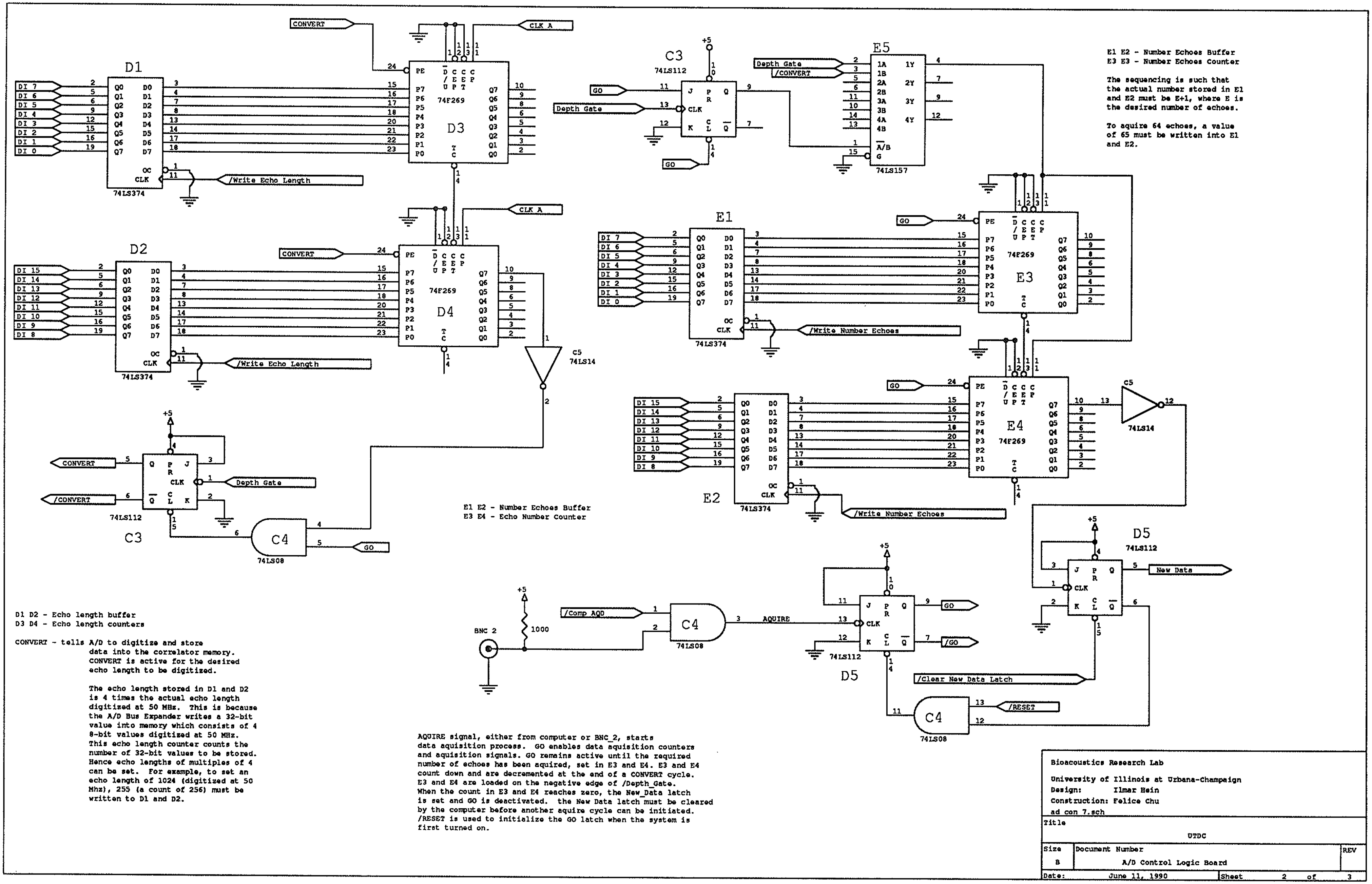
```
/******************************** STORE.C *************************************/
/*                                                                            */
/*   The echo acquisition program ACQUIRE stores echo data in the file        */
/*   MK500.ECH and the PLOTRES program writes the velocity plot into the      */
/*   file MK500.PLT.  This program renames the files MK500.ECH and MK500.PLT */
/*   to the files XX.ECH and XX.PLT where XX is the value in FILENUM.MK5. It */
/*   also increments the value in FILENUM.MK5.                                */
/*                                                                            */
/******************************************************************************/
#include <process.h>
#include <stdio.h>
#include <time.h>

main()
{
int
    result,
    fnum
    ;

char command_line [120],
     comments [120],
     fnme [15],
     fnumc[5],
     key
     ;

FILE *fptr;

result = system( "dir *.ech /w" );              /*  displays directory of .ech data files to screen */

       /* ---------- reads filunum to rename mk500.ech & mk500.plt to ---------- */
       if( ( fptr = fopen("filenum.mk5","r") ) == NULL )
         {
          printf("\n Unable to open FILENUM.MK5 file for reading...");
          exit(0);
         }
         fscanf(fptr,"%d", &fnum );
         fclose(fptr);
         fnum++;

       /* ---------- writes next file number to  filenum.mk5 ----------------- */
       if( ( fptr = fopen("filenum.mk5","w") ) == NULL )
         {
          printf("\n Unable to open FILENUM.MK5 file for writing...");
          exit(0);
         }
       fprintf(fptr, "%d",fnum);
       fclose(fptr);

       itoa( fnum, fnumc, 10 );
       strcpy(fnme,fnumc );
       printf("\nWriting to %s\n",fnme);
       strcpy(command_line, "ren mk500.ech ");         /* rename .ech file */
       strcat(command_line,fnme);
       strcat(command_line,".ech");
       printf("\n\n%s \n", command_line );
       result = system( command_line );

       strcpy(command_line, "ren mk500.plt ");         /* rename .plt file */
       strcat(command_line, fnme);
       strcat(command_line,".plt");
       printf("%s \n", command_line );
       result = system( command_line );                /* plot directory of .ech files to screen */
       result = system( "dir *.ech /w");
}
/********************************** end store.c *********************************************************/
```

APPENDIX F

A/D SUBSYSTEM SCHEMATICS AND TIMING DIAGRAMS



* See Chen, Jerome T., "Design and Implementation of a High-Speed Residue Number System Correlator for Ultrasonic Time Domain Blood Flow Measurement", Master's Degree Thesis in Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, 1990.



E1 E2 - Number Echoes Buffer
E3 E3 - Number Echoes Counter

The sequencing is such that the actual number stored in E1 and E2 must be E+1, where E is the desired number of echoes.

To aquire 64 echoes, a value of 65 must be written into E1 and E2.

E1 E2 - Number Echoes Buffer
E3 E4 - Echo Number Counter

D1 D2 - Echo length buffer
D3 D4 - Echo length counters

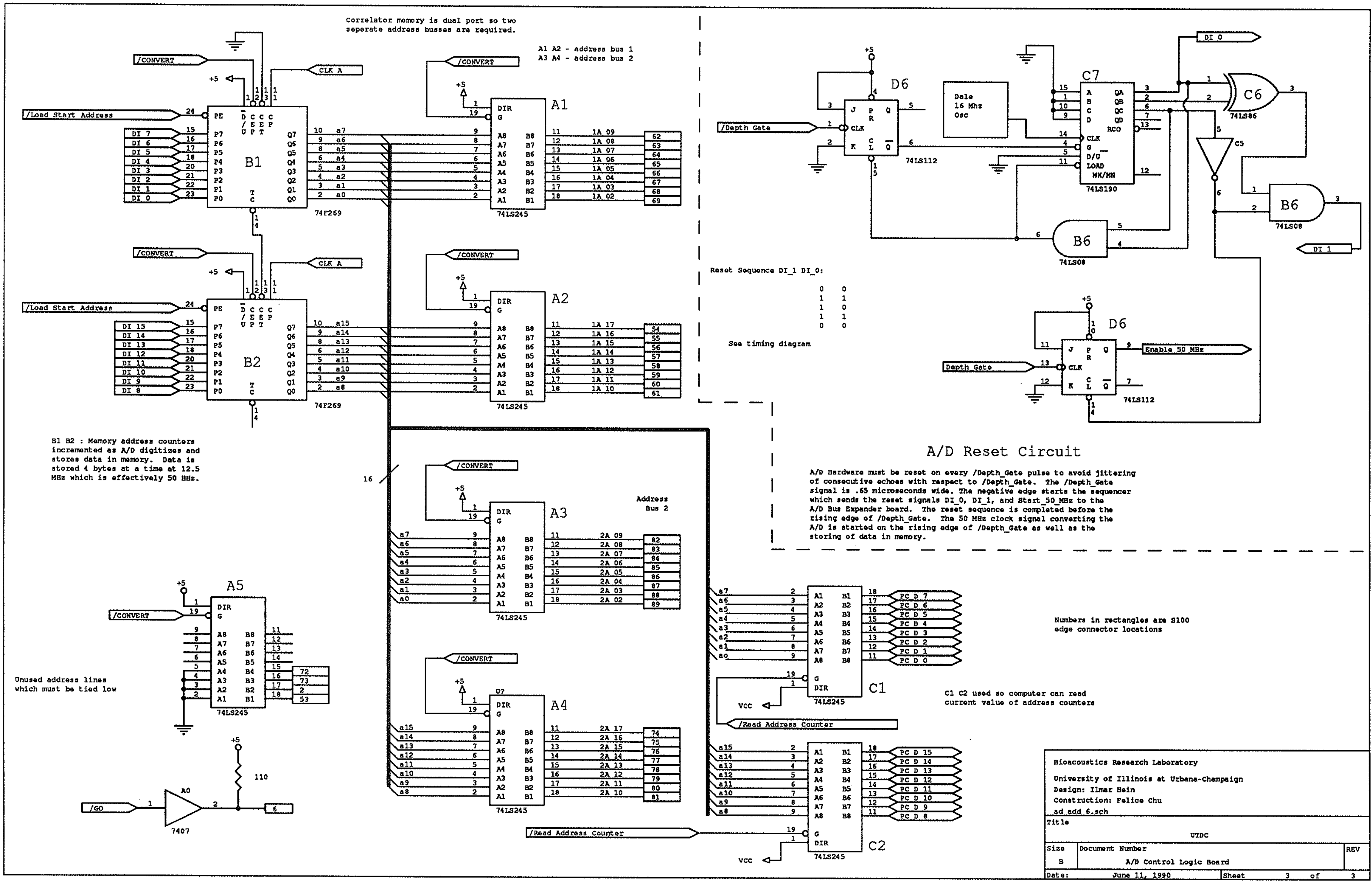
CONVERT - tells A/D to digitize and store data into the correlator memory. CONVERT is active for the desired echo length to be digitized.

The echo length stored in D1 and D2 is 4 times the actual echo length digitized at 50 MHz. This is because the A/D Bus Expander writes a 32-bit value into memory which consists of 4 8-bit values digitized at 50 MHz. This echo length counter counts the number of 32-bit values to be stored. Hence echo lengths of multiples of 4 can be set. For example, to set an echo length of 1024 (digitized at 50 Mhz), 255 (a count of 256) must be written to D1 and D2.

AQUIRE signal, either from computer or BNC_2, starts data aquisition process. GO enables data aquisition counters and aquisition signals. GO remains active until the required number of echoes has been aquired, set in E3 and E4. E3 and E4 count down and are decremented at the end of a CONVERT cycle. E3 and E4 are loaded on the negative edge of /Depth_Gate. When the count in E3 and E4 reaches zero, the New_Data latch is set and GO is deactivated. the New Data latch must be cleared by the computer before another aquire cycle can be initiated. /RESET is used to initialize the GO latch when the system is first turned on.

Bioacoustics Research Lab
University of Illinois at Urbana-Champaign
Design: Ilmar Hein
Construction: Felice Chu
ad con 7.sch

Title	UTDC		
Size B	Document Number A/D Control Logic Board		REV
Date:	June 11, 1990	Sheet 2	of 3



Correlator memory is dual port so two seperate address busses are required.

A1 A2 - address bus 1
A3 A4 - address bus 2

B1 B2 : Memory address counters incremented as A/D digitizes and stores data in memory. Data is stored 4 bytes at a time at 12.5 MHz which is effectively 50 HHz.

Unused address lines which must be tied low

Reset Sequence DI_1 DI_0:

0	0
1	1
1	0
1	1
0	0

See timing diagram

A/D Reset Circuit

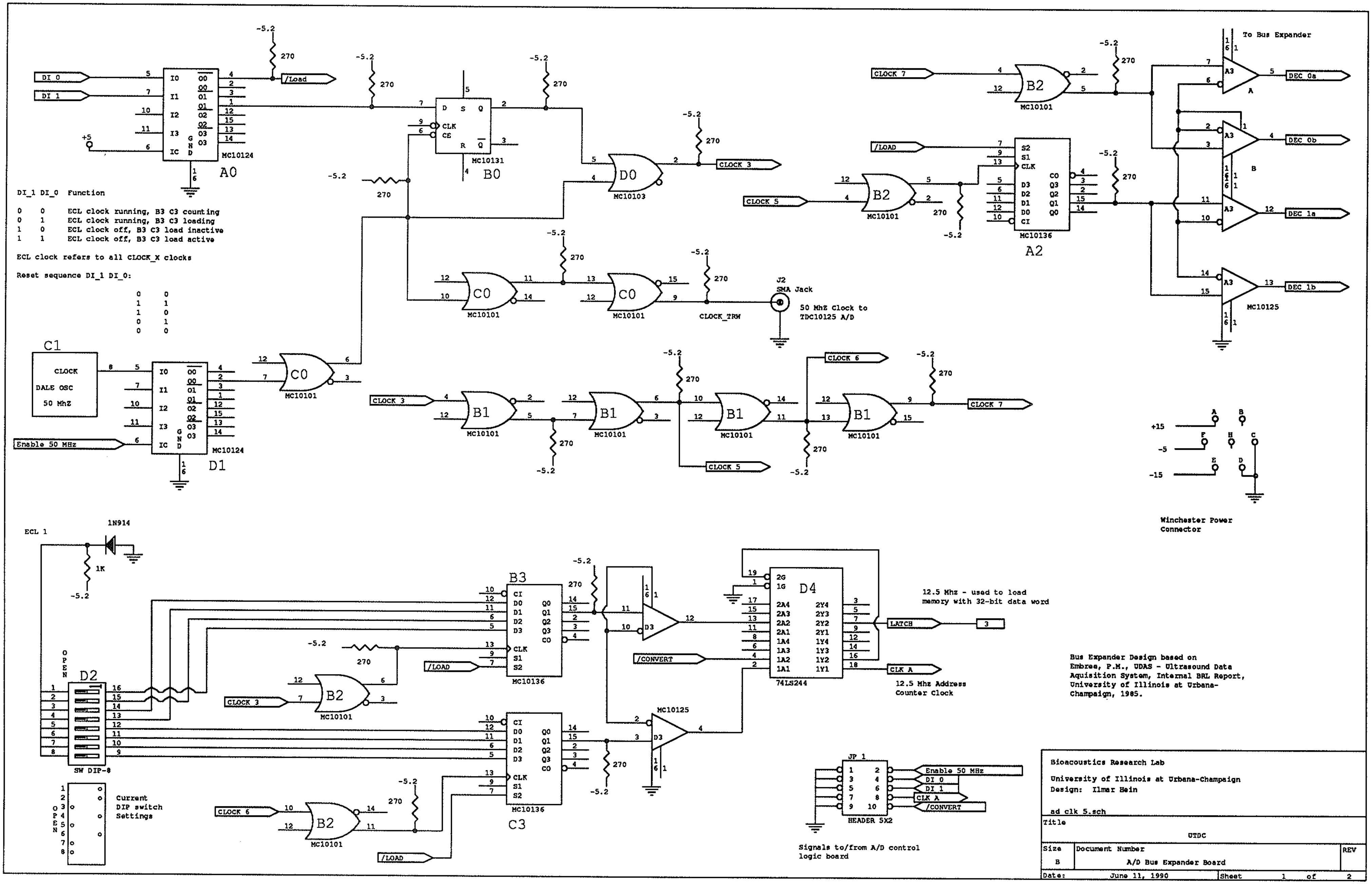
A/D Hardware must be reset on every /Depth_Gate pulse to avoid jittering of consecutive echoes with respect to /Depth_Gate. The /Depth_Gate signal is .65 microseconds wide. The negative edge starts the sequencer which sends the reset signals DI_0, DI_1, and Start_50_MHz to the A/D Bus Expander board. The reset sequence is completed before the rising edge of /Depth_Gate. The 50 MHz clock signal converting the A/D is started on the rising edge of /Depth_Gate as well as the storing of data in memory.

Numbers in rectangles are S100 edge connector locations

C1 C2 used so computer can read current value of address counters

Bioacoustics Research Laboratory
University of Illinois at Urbana-Champaign
Design: Ilmar Hein
Construction: Felice Chu
ad add 6.sch

Title	UTDC	
Size B	Document Number A/D Control Logic Board	REV
Date: June 11, 1990	Sheet 3 of 3	



DI_1	DI_0	Function
0	0	ECL clock running, B3 C3 counting
0	1	ECL clock running, B3 C3 loading
1	0	ECL clock off, B3 C3 load inactive
1	1	ECL clock off, B3 C3 load active

ECL clock refers to all CLOCK_X clocks

Reset sequence DI_1 DI_0:

```
0    0
1    1
1    0
0    1
0    0
```

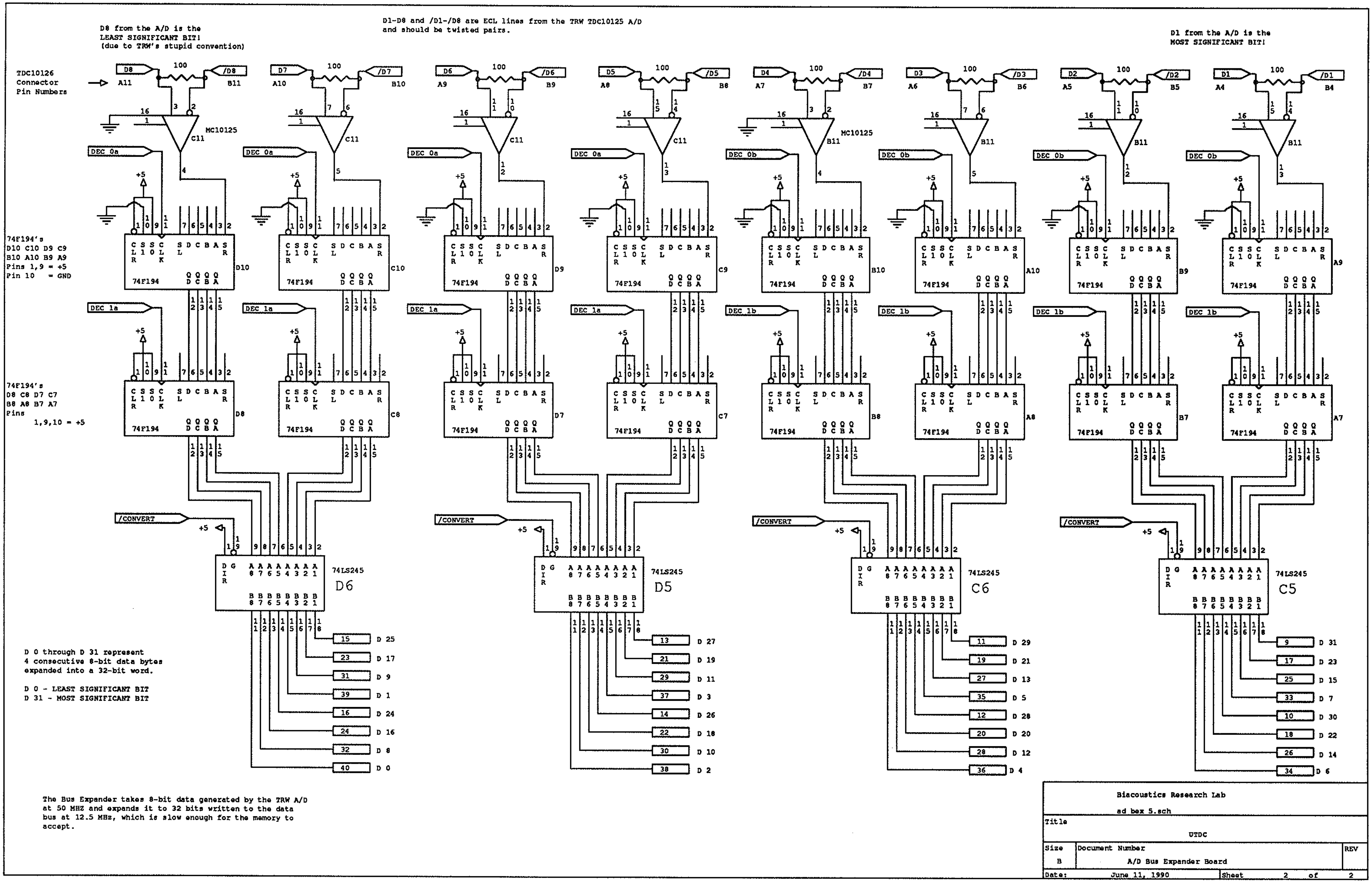
Bus Expander Design based on
Embree, P.M., UDAS - Ultrasound Data
Aquisition System, Internal BRL Report,
University of Illinois at Urbana-
Champaign, 1985.

Bioacoustics Research Lab

University of Illinois at Urbana-Champaign
Design: Ilmar Bein

ad clk 5.sch

Title	UTDC	
Size B	Document Number A/D Bus Expander Board	REV
Date: June 11, 1990	Sheet 1 of 2	



D8 from the A/D is the
LEAST SIGNIFICANT BIT!
(due to TRW's stupid convention)

D1-D8 and /D1-/D8 are ECL lines from the TRW TDC10125 A/D
and should be twisted pairs.

D1 from the A/D is the
MOST SIGNIFICANT BIT!

TDC10126
Connector
Pin Numbers

74F194's
D10 C10 D9 C9
B10 A10 B9 A9
Pins 1,9 = +5
Pin 10 = GND

74F194's
D8 C8 D7 C7
B8 A8 B7 A7
Pins
1,9,10 = +5

D 0 through D 31 represent
4 consecutive 8-bit data bytes
expanded into a 32-bit word.

D 0 - LEAST SIGNIFICANT BIT
D 31 - MOST SIGNIFICANT BIT

The Bus Expander takes 8-bit data generated by the TRW A/D
at 50 MHZ and expands it to 32 bits written to the data
bus at 12.5 MHz, which is slow enough for the memory to
accept.

Biacoustics Research Lab

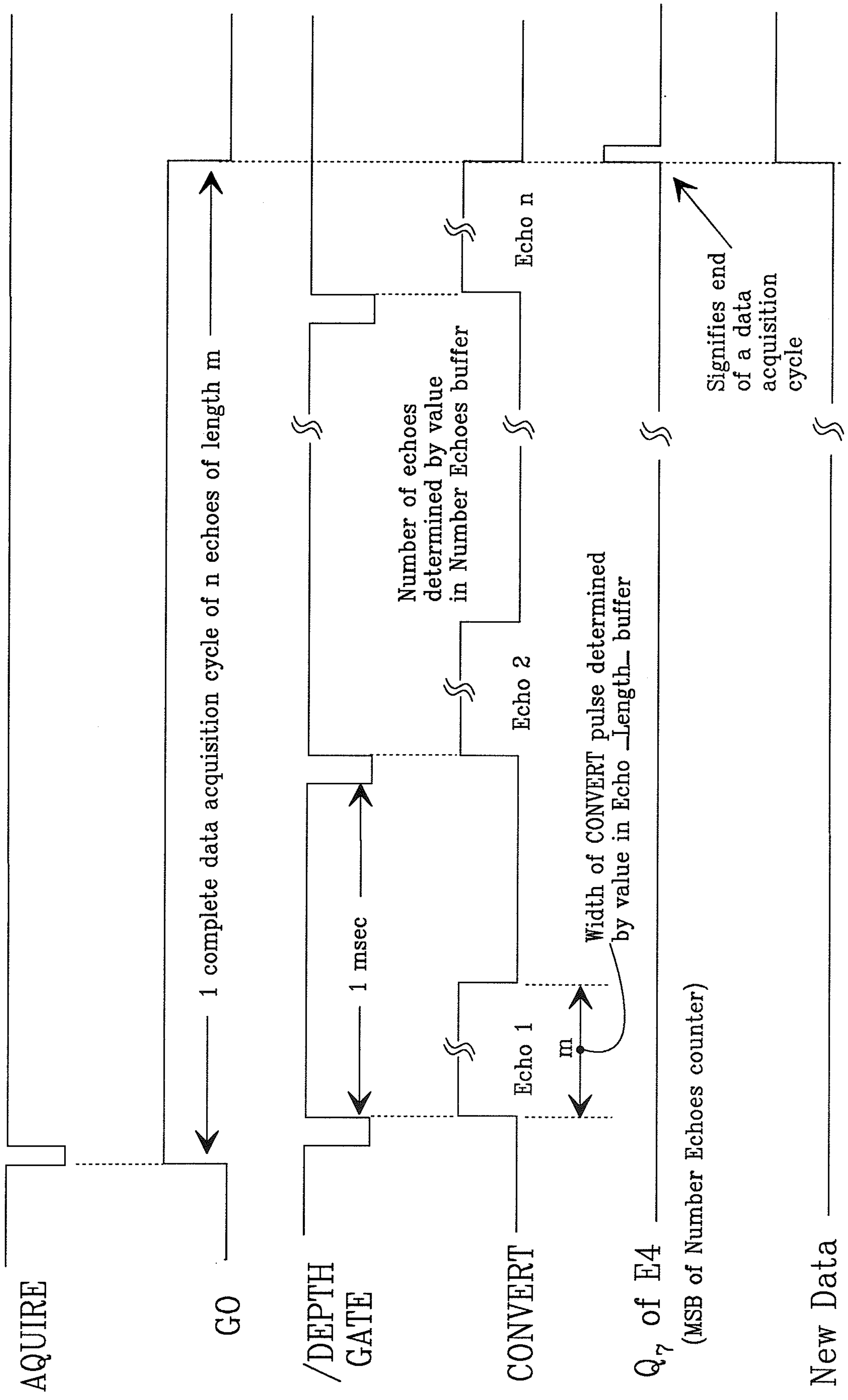
ad bex 5.sch

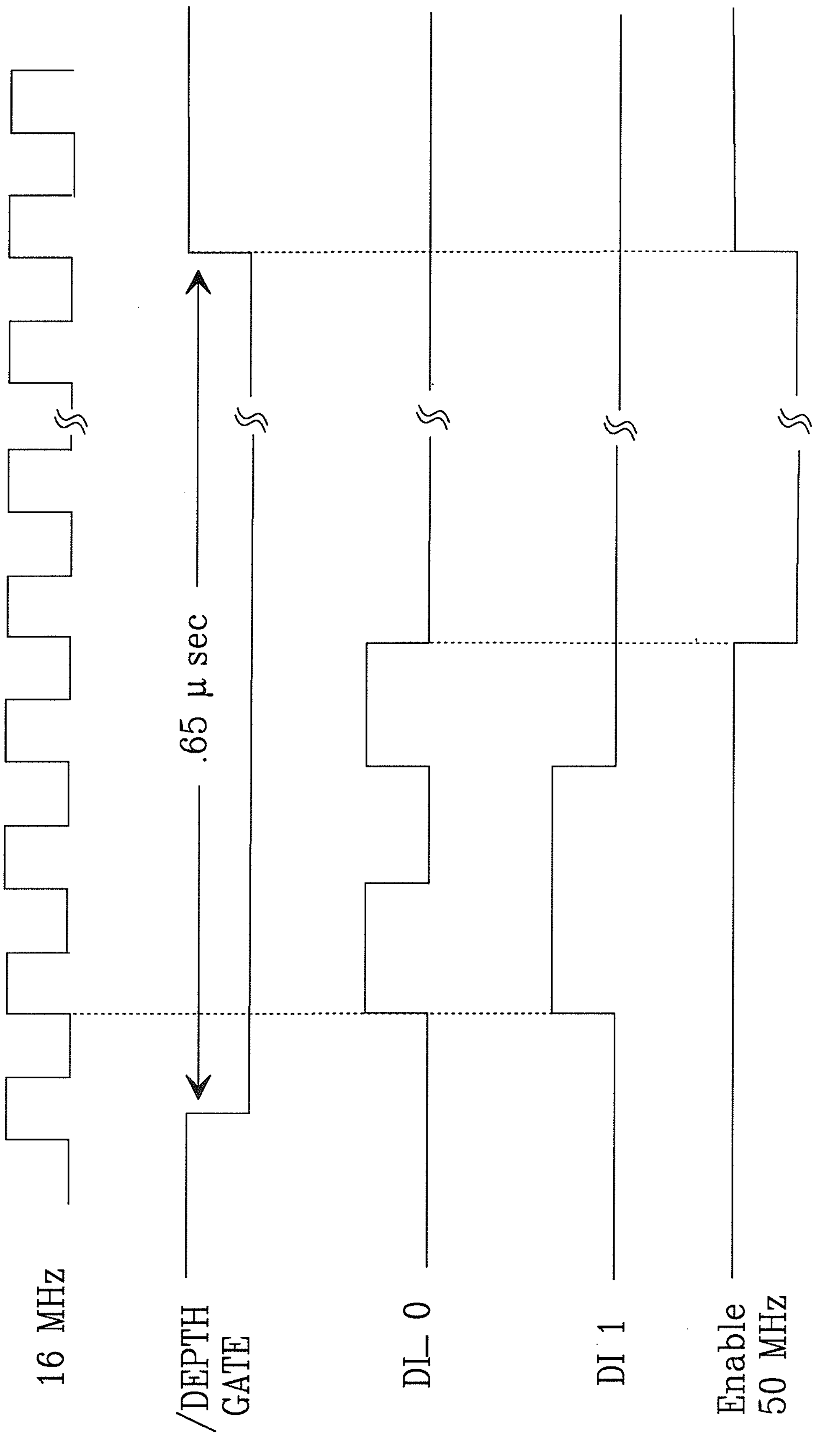
Title

UTDC

Size	Document Number	REV
B	A/D Bus Expander Board	

Date:	June 11, 1990	Sheet	2	of	2


AQUIRE
GO
/DEPTH GATE
CONVERT
Q_7 of E4
(MSB of Number Echoes counter)
New Data
1 complete data acquisition cycle of n echoes of length m
1 msec
Echo 1
m
Width of CONVERT pulse determined by value in Echo_Length_ buffer
Echo 2
Number of echoes determined by value in Number Echoes buffer
Echo n
Signifies end of a data acquisition cycle


16 MHz
/DEPTH GATE
DI_0
DI 1
Enable 50 MHz
.65 µ sec

APPENDIX G

UTDC BLOOD FLOWMETER PROGRAM

```
/************************************************************************************************************/
/*                                                                                                          */
/*                                            UTDC Blood Flowmeter                                           */
/*                                                                                                          */
/*                                                  8-19-90                                                 */
/*                                                                                                          */
/*                                                                                                          */
/*    This program is used with the UDA-RNC system and the ATL MK500 imager.  It has functions that test    */
/*    various aspects of the A/D, it can display digitized RF echoes from the ATL imager, and it can plot   */
/*    the velocity vs range using the Ultrasound Time Domain Correlation (UTDC) technique.  The velocity    */
/*    plots as well as the RF data can be stored to disk.  The RF data can be loaded back into the UDA-RNC  */
/*    system and processed with different parameters.                                                       */
/*                                                                                                          */
/*    This program was compiled with Borland Turbo-C                                                        */
/*                                                                                                          */
/************************************************************************************************************/

#include <conio.h>
#include <stdio.h>
#include <string.h>
#include <graphics.h>
#include <time.h>
#include <math.h>
#include <dos.h>

/* The following are PC port addresses that communicate with UDA-RNS */

#define CONTROL               0x310    /* determines whether correlator or A/D has control of the     */
                                       /* UDA-RNC's data bus.  Writing a 5 to CONTROL gives the       */
                                       /* correlator contro; writing a 0 gives A/D control            */

/* The following are A/D function address.  They are only valid when CONTROL is passed to the A/D    */

#define RESET                 0x300    /* Reset A/D latches                                            */
#define COMP_ACQUIRE_DATA      0x302   /* Tells RNS_UDAS to start an acquire data cycle                */
#define WRITE_ECHO_LENGTH     0x304    /* Echo length divided by 4 stored here                         */
#define WRITE_NUM_ECHOES      0x306    /* number of consecutive echoes to acquire is stored here       */
#define CLEAR_ND_LATCH        0x308    /* Clears the new data latch                                    */
#define LOAD_START_ADDRESS    0x30a    /* loads start address of memory where echoes are to be placed */
#define READ_ADDRESS_COUNTER  0x30a    /* Read the current value of the memory address counter         */
#define RD_TRANS_POS          0x30c    /* Read the current transducer position.  Bits 0-6 are the      */
                                       /* position of the wiper, bits 7 is the New Data latch, and     */
                                       /* and bits 8-15 are the wiper location.                        */


/* The following are addresses that involve correlator functions.  These are active only when        */
/* CONTROL is passed to the correlator.  The address values are the same as above but are labelled   */
/* differently to distinguish between correlator and A/D functions                                    */

#define CORR_ADD_REG_1_LO     0x300    /* Load lower 12 bits of correlator addres register 1           */
#define CORR_ADD_REG_2_LO     0x302    /* Load lower 12 bits of correlator addres register 2           */
#define CORR_ADD_REGS_HI      0x304    /* Load upper 8 bits of correlator address registers 1 and 2    */
#define CORR_DATA_REG_12      0x306    /* Load correlator data registers 1 and 2                       */
#define CORR_DATA_REG_34      0x308    /* Load correlator data registers 3 and 4                       */
#define CORR_READ_MEM         0x30a    /* Read data from correlator main memory                        */
#define CORR_WRITE_MEM        0x312    /* Write data to correlator main memory                         */


#define STP gotoxy          /* Alias for Set Text Position */
#define CLS cleardevice     /* Clears the screen           */

#define MAX_CLIP 479        /* Maximum and minimum y-values for the VGA display, which is          */
#define MIN_CLIP 0          /* 640 x 480.  Numbers above and below will be clipped to these values */
#define X_AXIS   350        /* VGA y-value where x=0 axis is placed                                */
```

```
        char key,              /* used to get key the from keyboard       */
             datafile [15],    /* datafile name                           */
             acquire_time[15], /* stores time of acquisition              */
             txt[80],          /* used in printing text in graphics mode  */
             choice
             ;

         int
             effective_pr_period = 1,  /* effective pulse repetition period                  */
             bank = 1,                 /* determines which memory bank is used in echo display */
             old_flow[100],            /* used to erase previous flow trace                  */
             set_av = 1,               /* echo sets to average (blocks of 20)                */
             base_echo = 0,            /* base echo where data will be stored at             */
             max_delta=10,             /* maximum delta correlations will be performed for   */
             res_fac = 1,              /* resolution factor                                  */
             range_inc = 40,           /* range increment for doing correlations             */
             xmax,                     /* number velocity values that will be plotted        */
             y_pt,                     /* variable used in plotting y value of flow          */
             num_rf_ech = 255,         /* number of RF echoes to store to disk               */
             number_echoes,            /* number echoes to be acquired                       */
             sav_flag = -1,            /* save RF flag, -1=no 1=yes                          */
             store_flag = -1,          /* store shift -1 = don't store, 1 = store            */
             load_flag = -1            /* load RF from disk to correlatore                   */
             ;

unsigned int c,                        /* counter                                 */
             d,                        /* counter                                 */
             xoffset,                  /* used in plotting                        */
             yoffset,                  /* used in plotting                        */
             ekos_to_plot = 1,         /* number of rf echoes to plot             */
             top8                      /* used in calculating correlator memory address */
             ;

unsigned int value,                   /* general purpose variable                  */
             values [1024]            /* used to read/write 1K echoes from/to memory */
             ;

unsigned long int address,                  /* used in writing address values to memory  */
                  display_echoes = 262144,  /* echo address variable, used in displaying */
                  msa                       /* starting address for a given echo         */
                  ;

     long
           zzz                              /* general purpose variable                  */
           ;
    float
           pass=0,           /* used in echo-length test routine to count passes  */
           flow[100],        /* shift values returned from correlation routine    */
           maxflow,          /* used to find maximum shift value in a plot        */
           maxvel,           /* maximum velocity measured in a plot               */
           xscale,           /* factor in plotting x value of flow                */
           vel_scale = 1.5 , /* no angle correction                               */
           yscale = 70.0,    /* factor in plotting y value of flow                */
           error_count=0     /* used to count errors in echo-length test routine  */
           ;


      int             /* Turbo-C graphics intialization variables */
           g_driver,
           g_mode,
           g_error
           ;


     long t;
     char *ptr;

    FILE
          *fptr
          ;
```

```
main()

{

void plot_flow();            /* does correlations and plots un-angle corrected velocity */
void plot_echo();            /* plots RF echoes to the screen                           */
void save_echoes();          /* writes RF echoes from memory to disk                    */
void load_echoes();          /* reads  RF data stored on disk to memory                 */
void corr (set_av, res_fac); /* correlation algorithm                                   */

outport(CONTROL,15);    /* Default should be correlator in control */


/*-------------------- initialize A/D -------------*/

outport(CONTROL, 0 );
outport(RESET, 0 );
outport( WRITE_NUM_ECHOES, 65 );
outport( WRITE_ECHO_LENGTH, 255 );
outport(CONTROL,15);


/*--------------- initialize turbo-C graphic modes--------*/

  detectgraph(&g_driver, &g_mode);
  initgraph(&g_driver, &g_mode, "");
  g_error = graphresult();

  if(g_error < 0)
    {
     printf("initgraph error: %s.\n", grapherrormsg(g_error));
     exit(1);
    }

    display_echoes = 1024 * (long)ekos_to_plot;



/*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ Main menu loop @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/

while(1)
  {

   initgraph(&g_driver, &g_mode, "");
   closegraph();
   textmode(3);



   /*---------------------------------------- MAIN MENU ----------------------------------------------------*/
   printf("\n***** Ultrasound Time Domain Correlation Flow Measurement System ****\n\n");

   printf("\n   [1] Write to address           [6] Write to Echo length latch");
   printf("\n   [2] Read from an address       [7] Write to # echoes latch");
   printf("\n   [3] Write infinite loop        [8] Read Address Ctr");
   printf("\n   [4] Write to an address & read [9] Rd TRans Pos");
   printf("\n   [5] Write address ctr & read   [a] Acquire Data ( Comp AQD ) ");
   printf("\n   [i] Initialize a/d             [c] Clear New Data Latch");
   printf("\n   [d] display RF echoes          [z] test echo length/ # echoes");
   printf("\n   [m] write & read from Memory   [q] Quit ");
   printf("\n   [f] plot Flow vs distance      [b] choose mem Bank (%d)", bank);
   printf("\n   [s] Sets to average (%2d)       [y] base echo (%d) ", set_av, base_echo );
   printf("\n   [r] Resolution factor   (%2d)   [w] Delta Max (%2d)", res_fac, max_delta );
   printf("\n     [e]  Echoes  to  plot  (%2d)             [p]  effective  PR  period  (%2d)",  ekos_to_plot,
effective_pr_period);
   printf("\n   [S] Save RF echo (%2d)          [L] Load RF (%2d)", sav_flag, load_flag);
   printf("\n   [N] # RF echoes to save (%3d)                 ", num_rf_ech );

   printf("\n\n   [g] sets to average = 20  delta max = 1  # echoes = 255");

   printf("\n\nWhaddaya wanna do? ");
```

```
fflush(stdin);
choice = getch();
switch( choice )
      {

       /*--------- g:  sets to average = 20  delta max = 1  # echoes = 255 ---------*/
        case 'g':
            max_delta = 1 ;
            set_av    = 20;
            outport(CONTROL, 0);
            outport(RESET, 0 );
            outport( WRITE_NUM_ECHOES, 255 );
            outport(CONTROL,15);
            break;


       /*---------- w:  set delate max -------------*/
        case 'w':
            printf("\nEnter Delta Max (1-10) " );
            scanf("%d",&max_delta);
            if( max_delta < 1 || max_delta > 20 )
              {
                printf("\n Requested input must fall between 1 and 10.\n");
                res_fac=1;
              }
            break;


       /*---------- p:  set effective pulse repetietion period -------------*/
        case 'p':
            printf("\nEffective PR Period (1-10): " );
            scanf("%d",&effective_pr_period);
            if( effective_pr_period < 1 || effective_pr_period >10)
              {
                printf("\n Damn it!  It's gotta be between  1 and 10.\n");
                effective_pr_period = 1;
              }
            break;



       /*---------- N:  set Number of RF echoes to save to disk -------------*/
        case 'N':
            printf("\nNumber RF echoes to save (1-255): " );
            scanf("%d",&num_rf_ech);
            if( num_rf_ech < 1 || num_rf_ech > 255)
              {
                printf("\n No way, dude... gotta be between  1 and 255.\n");
                num_rf_ech = 1;
              }
            break;


       /*--------- r: set resolution factor ------------*/
        case 'r':
            printf("\nEnter Resolution Factor: [1] .6 mm, [2] .3 mm, or [3] .15 mm >" );
            scanf("%d",&res_fac);
            if( res_fac < 1 || res_fac > 3 )
              {
                printf("\n Requested input must fall between 1 and 3.\n");
                res_fac=1;
              }
            break;
```

```
/*---------- y:  set base echo ------------*/
case 'y':
    printf("\nEnter Base Echo: " );
    scanf("%d",&base_echo);
    if( base_echo < 1 || base_echo > 200 )
      {
        printf("\n Hey dude, like, between 1 and 200 would be really cool!!\n");
        base_echo = 0;
      }
    break;


/*---------- s:  set number of 20-echo sets to average ------------*/
case 's':
    printf("\nEnter Sets to average ( 1 <= S <= 24 ): " );
    scanf("%d",&set_av);

    if( set_av < 1 || set_av > 24 )
      {
        printf("\n Hey man, BETWEEN 1 and 24!!\n");
        set_av = 1;
      }

    number_echoes = set_av * 10 + 15;
    outport(CONTROL, 0 );
    outport( WRITE_NUM_ECHOES, number_echoes );
    outport(CONTROL,15);
    break;


/------- b:  set memory bank used in displaying and writing data to disk -----*/
case 'b':
   bank = bank * -1;
   break;

/*------ S:  set save-RF-data-to-disk flag ------------*/
case 'S':                          /* sav_flag=1, save */
   sav_flag = sav_flag * -1;
   break;


/*---------- L:  Load RF data from disk to corr memory------*/
case 'L':                          /* load_flag = 1, load    */
   load_flag = load_flag * -1;      /* load_flag =-1, acquire */
   if( load_flag == 1 )
     {
      CLS;
      puts("Loading RF data into correlator...");
      load_echoes();
     }
   break;


/*---------- e:  number of echoes from memory to display ------------*/
case 'e':
    printf("Enter echoes to be displayed:");
    scanf("%d",&ekos_to_plot);
    display_echoes = 1024 * (long)ekos_to_plot;
    printf("%lu\n", display_echoes );
    break;

/*---------- 1:  write value to A/D address buffer ------------*/
case '1':
     printf("\n\n Enter address (hex)  value : ");
     fflush(stdin);
     scanf("%x %x", &address, &value);
     printf("address= %x ", address);
                    printf("value = %x\n", value );
     outport( address, value );
     break;
```

```
  /*---------- 2:  read value from A/D address buffer -------------*/
  case '2':

       printf("\n\nEnter address to read:  ");
       fflush(stdin);
       scanf("%x", &address);
       value = inport(address);
       printf("Address %x contains %x", address, value);
       break;


  /*---------- 3:  write continously to A/D address buffer -------------*/
  case '3':
       printf("\n\nEnter address (hex)   value: ");
       fflush(stdin);
       scanf("%x %x", &address, &value );
       while( !kbhit() )
            outport(address, value);
       break;


  /*---------- 4:  write to an A/D address and read back  -------------*/
  case '4':
      printf("\n\nEnter address(hex) value: ");
      fflush(stdin);
      scanf("%x %x",&address, &value );
      outport(address, value);
      value = inport(address);
      printf("\nAddress = %x  value = %x  %d", address, value, value);
      fflush(stdin);
      break;


  /*---------- 5:  write to address counter & read back -------------*/
  case '5':
    printf("Enter value to write: ");
    fflush(stdin);
    scanf("%x", &value);
    outport(LOAD_START_ADDRESS, value);
    value = 0;
    value = inport(READ_ADDRESS_COUNTER);
    printf("\nValue returned = %xh  %d", value,value);
    break;


 /*---------- 6: write to a/d echo length latch  -------------*/
 case '6':
    printf("\Enter value to write to Echo Length Ctr: ");
    fflush(stdin);
    scanf("%d", &value);
    outport(CONTROL, 0);
    outport( WRITE_ECHO_LENGTH, value );
    outport( CONTROL, 5 );
    break;


 /*---------- 7:  write to a/d # echoes latch -------------*/
 case '7':
   printf("\nEnter value to write to # Echoes latch: ");
   fflush(stdin);
   scanf("%d", &value);
   outport(CONTROL, 0);
   outport( WRITE_NUM_ECHOES, value );
   outport(CONTROL, 5);
   break;

/*---------- 8:  read address counter ------------*/
case '8':
  value = inport( READ_ADDRESS_COUNTER );
  printf("\nAddress Counter = %xh %d ",value,value );
  break;
 /*---------- 9:  read transducer position -------------*/
case '9':
while(1)
 {
  value = inp (RD_TRANS_POS);
  printf("%x\n", value &0x0080);
 }
break;


/*---------- a:  send acquire data signal -------------*/
case 'a':
   outport(COMP_Acquire_DATA, 0 );
   break;


/*---------- c:  send acquire data signal -------------*/
case 'c':
  outport(CLEAR_ND_LATCH, 0 );
  break;


/*---------- i:  initialize a/d -------------*/
case 'i':
     outport(CONTROL, 0 );
     outport(RESET, 0 );
     outport( WRITE_NUM_ECHOES, 65 );
     outport( WRITE_ECHO_LENGTH, 255 );
     outport(CONTROL,15);
     break;


 /*---------- m:  write & read to correlator memory -------------*/
 case 'm' :
   while (1)
     {
      printf("\nEnter value to write: ");
      scanf("%ld", &zzz);
      if( zzz > 255 )break;

      for( c=0; c<640; c++ )
          {
           outport(CONTROL, 15 );
           outport(CORR_ADD_REG_1_LO,c);
           outport(CORR_ADD_REG_2_LO,c);
           outport(CORR_ADD_REGS_HI,0);
           zzz = (long)c;
           if(zzz >255) zzz = (long)(c - 256);
           else if( zzz > 511 ) zzz = (long)(c - 512);

           outport(CORR_DATA_REG_12, zzz);
           outport(CORR_DATA_REG_34, zzz);
           outport(CORR_WRITE_MEM,0);
          }
      outport( CONTROL, 5 );
      outport( CORR_ADD_REG_1_LO, 0 );
      outport( CORR_ADD_REG_2_LO, 0 );
      outport( CORR_ADD_REGS_HI,  0 );
      zzz = inport( CORR_READ_MEM );
      printf("\nValue returned = %lx \n",zzz );
     }
     outport(CONTROL,0);
     break;


/*---------- q:  exit program -------------*/
case 'q':
    textmode(3);
    exit(0);
    break;
```

```
        /*---------- d:  display RF echoes -------------*/
        case 'd':
            plot_echo();
            break;


        /*---------- f:  plot flow velocity vs range -------------*/
        case 'f':
            plot_flow();
            break;




       /*------------------- Case z -----------------------*/
       /*                                                  */
       /* Test routine to check if Echo Length counter and */
       /* Number Echoes counter are couning correctly.     */
       /* Assumes a value of 255 (Echo Length of 256)  has */
       /* been written into the Echo Length counter and a  */
       /* value of 65 (64 echoes) has been written to the  */
       /* Number Echoes counter.  After acquiring data, the */
       /* address counter should contain 0x4000.          */
       /*                                                  */
       /*--------------------------------------------------*/
       case 'z':
            outport(CONTROL,0);
            while(1)
                 {
                  outport( LOAD_START_ADDRESS, 0);
                  outport( CLEAR_ND_LATCH, 0 );
                  outport( COMP_Acquire_DATA, 0 );
                  while( ( inport(RD_TRANS_POS) & 0x0080 ) == 0 ) ;
                  value = inport(RD_TRANS_POS);

                  pass++;

                  value = inport(READ_ADDRESS_COUNTER);
                  if(value != 0x4000)
                    {
                     error_count++;
                     printf("\n***pass = %f  error_count = %f  value = %x",pass,error_count,value);
                    }

                    printf("\npass = %f   error_count = %f  value = %x",pass, error_count, value);
                 }
             outport(CONTROL,15);
             break;

        default:
             ;

       } /* end switch */
  }
  /*@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@ end main menu loop @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@*/


}
/***************************** end main **********************************************************/
```

```
/************************************ plot_flow *****************************************************/
/*                                                                                                  */
/*  This routine plots the flow vs range.  It gets the time shifts from the correlator and         */
/*  calculates and plots the non-angle corrected flow.                                              */
/*                                                                                                  */
/*  options available while plotting:                                                               */
/*                                                                                                  */
/*      [1] set vertical scaling to 0.2 cm/sec                                                      */
/*      [2]  "      "       "       0.8   "                                                         */
/*      [3]  "      "       "       1.5   "                                                         */
/*      [4]  "      "       "       2.0   "                                                         */
/*                                                                                                  */
/*      [A]  preset parameters to max_delta = 1                                                     */
/*                                set_av    = 20                                                    */
/*                                                                                                  */
/*      [B]  preset parameters to max_delta = 10                                                    */
/*                                set_av    = 1                                                     */
/*                                                                                                  */
/*      [C]  preset parameters to max_delta = 10                                                    */
/*                                set_av    = 3                                                     */
/*                                                                                                  */
/*      [c]  enter comment into recorded shift data file                                            */
/*                                                                                                  */
/*      [d]  increment max delta                                                                    */
/*      [D]  decrement  "   "                                                                       */
/*                                                                                                  */
/*      [q]  quit                                                                                   */
/*                                                                                                  */
/*      [p]  increment effective PR period                                                          */
/*      [P]  decrement    "      "    "                                                             */
/*                                                                                                  */
/*      [s]  increment sets to average                                                              */
/*      [S]  decrement  "   "    "                                                                  */
/*                                                                                                  */
/*      [r]  record plot data (shifts, not vel) to disk                                             */
/*                                                                                                  */
/*/*************************************************************************************************/

 void plot_flow()
 {

 initgraph(&g_driver, &g_mode, "");  /* initialize graphics */
 CLS;

 /*--------------initialize vertical scaling ---------------*/
 yscale = 46.67;
 STP(8,1);
 strcpy(txt," VERT = 1.5 cm/sec / div");
 printf("%s",txt);
 STP(2,1);
 strcpy(txt,"7.5");
 printf("%s", txt);

 for(c=0; c<100; c++) old_flow[c]=0;     /* initialize buffer used in erasing the flow trace            */

 xscale = 25;                            /* initialize horizontal scaling                               */
 xmax = 23;

 /*++++++++++++++++++++++++++++++++ loop until q key hit ++++++++++++++++++++++++++++++++++++++*/
 while(1)
  {


     key = '0';

     if( load_flag == -1 )      /* need to acquire data before correlating */
       {
```

```
/* ----------------------- load start address ----------------------------------------------------*/
while( (value = inport(READ_ADDRESS_COUNTER) ) != 0 )     /* clock is reset on next depth gate pulse    */
      {

       outport(CONTROL, 0 );
       outport(LOAD_START_ADDRESS, 0);            /* from the ATL. The computer must wait until that */
       STP(55,4);                                 /* pulse arrives. This is done by writing values to */
       sprintf(txt,"START Address = %4x",value );  /* the address counter... when the value has been  */
       printf("%s",txt);                          /* written in , we know that the clock has         */
       STP(55,5);                                 /* been started.                                   */
       strcpy(txt,"END Adress = waiting  ");
       sprintf("%s",txt );

       if( kbhit() )
         if( ( key = getch() ) == 'q' ) return;
      }

STP(55,5);
strcpy(txt,"END Address = loaded ");
printf("%s",txt );

STP(55,4);
sprintf(txt,"START Address = %4x",value );
printf("%s",txt);

/*----------------- acquire data ------------------------------------------------------------------*/
outport(CLEAR_ND_LATCH, 0 );                       /* Initialize  New Data latch to 0                  */
outport(COMP_ACQUIRE_DATA, 0 );
while(  ( inport( RD_TRANS_POS ) & 0x0080 ) == 1 )  /* Wait til New Data latch cleared                  */
        if( kbhit() )
           if( ( key = getch() ) == 'q' ) return;

STP(55,3);
strcpy(txt,"ACQUIRING DATA");
printf("%s",txt);
while(  ( inport( RD_TRANS_POS ) & 0x0080 ) == 0 )  /* Wait til New Data latch = 1, indicating acquire */
       if( kbhit() )                                 /* data cycle is complete                         */
         if( ( key = getch() ) == 'q' ) return;

value = inport( READ_ADDRESS_COUNTER );
STP(55,5);
sprintf(txt,"END Address = %4x   ",value );
printf("%s",txt);
}

/*--------- print current acquisition parameters to screen ---------------------------------*/
STP(40,25);
sprintf(txt,"EPRP = %2d  sets = %2d  delta = %2d", effective_pr_period, set_av, max_delta);
printf("%s",txt);

/*-------------- correlation   fuctions ---------------------------------------------------*/
STP(55,3);
strcpy(txt,"Correlating  ");
printf("%s",txt);

outport( CONTROL, 5 );          /* pass control of RNS_UDAS bus from A/D to correlator */

time(&t);                      /* print correlation  time to screen */
ptr = asctime(localtime(&t));
*(ptr + 20) = 0;
strcpy(acquire_time, ptr+11);
STP(70,1);
printf( "%s", acquire_time );

if( load_flag == 1 )           /* Are correlations done on stored data?      */
   {
   STP(55,3);
   sprintf(txt,"Stored Data");
   printf("%s",txt);
   }
```

```
  corr( set_av , res_fac);      /* do correlations */
  STP(55,3);
  strcpy(txt,"             ");
  printf("%s",txt);


  /*------------------ save SHIFT (not velocity) information to disk? Also find max shift. -----------*/
  maxflow = 0.0;
  if( store_flag==1 )
    {
     fprintf(fptr, "\n%s\n", acquire_time );
    }
                                         /* also calculate max flow in this loop */
  maxflow = 0.0;
  for(c=0; c<=xmax; c++)
      {
       if( flow[c] > maxflow) maxflow = flow[c];
       if( store_flag == 1 ) fprintf(fptr,"%f %f \n",(float)c * .6/(float)res_fac, flow[c] );
      }


  /*----------- calculate and print the maximum velocity of this plot -------------------------*/
  maxvel = (vel_scale / effective_pr_period) * maxflow;
  STP( 43,1 );
  sprintf(txt,"Max Vel = %.2f", maxvel );
  printf("%s", txt);


  /*------------ erase the previous trace -----------------------------------------------------*/
  setcolor(0);
  moveto( 0,(int) old_flow[0] );
  for(c=1; c<=xmax; c++)
      {
       lineto( (int)((float)c * xscale),  old_flow [c] );
      }


 /*--------------------- plot the horizontal axis --------------------------------------------*/
 setcolor(LIGHTGREEN);
 moveto(0,350);      /* horizontal axis */
 lineto(600,350);
 STP(1,24);
 if( res_fac == 1)
  strcpy(txt,"      1    2    3    4     5    6    7    8     9   10   11   12   13   14   ");
 if( res_fac == 2)
  strcpy(txt,"           1         2          3         4        5          6              ");
 if( res_fac == 3)
  strcpy(txt,"                     1                    2                   3              ");

 printf("%s",txt);
 STP(25,25);
 strcpy(txt,"Distance (mm)");
 printf("%s",txt);

 for(c=0; c<15; c++)
     {
      moveto( (int)( (float)c*41.67 ), 345 );
      lineto( (int)( (float)c*41.67 ), 355 );
     }


/*--------------------- plot the vertical axis ----------------------------------------------*/
moveto(0,0);    /* vertical axis */
lineto(0,350);

for( c=0; c<=350; c=c+70 )
   {
    moveto(0,c);
    lineto(5,c);
   }
 if( key == '0' ) if( kbhit() ) key = getch();


/*====================== velocity plotting options ====================================*/
if( key !='0' )
  {
   STP(1,25);
   sprintf(txt,"%c", key);
   printf("%s",txt);

   STP(8,1);

   /*---------------- 4:  change vertical scale to 2 cm/sec ---------------------*/
   if( key == '4' )
     {
      yscale = 35.00;
      strcpy(txt, " VERT = 2 cm/sec / div ");
      printf("%s",txt);

      STP(2,1);
      strcpy(txt,"10 ");
      printf("%s",txt);
     }

   /*---------------- 3:  change vertical scale to 1.5 cm/sec ---------------------*/
   if( key == '3' )
     {
      yscale = 46.67;
      strcpy(txt, " VERT = 1.5 cm/sec / div ");
      printf("%s",txt);

      STP(2,1);
      strcpy(txt,"7.5");
      printf("%s",txt);
     }

   /*---------------- 2:  change vertical scale to 0.8 cm/sec ---------------------*/
   if( key == '2' )
     {
      yscale = 87.5;
      strcpy(txt," VERT =  .8 cm/sec / div ");
      printf("%s",txt);
      STP(2,1);
      strcpy(txt,"4.0");
      printf("%s",txt);
     }

   /*---------------- 1:  change vertical scale to 0.2 cm/sec ---------------------*/
   if( key == '1' )
     {
      yscale = 350.0;
      printf(" VERT =  .2 cm/sec / div ");
      STP(2,1);
      strcpy(txt,"1.0");
      printf("%s", txt );
     }


   /*-------------------------- r: record data to disk (toggles )------------------*/
   if( key == 'r')
     {
      store_flag = store_flag * -1;
      if( store_flag == 1 )                                           /* record */
        {

         time(&t);
         ptr = asctime(localtime(&t));

         STP(  10, 10 );                                   /* synchronize to tape */
      strcpy(txt,"Enter tape counter value from VCR: "); /* counter             */
      printf("%s", txt);
      gets( datafile );
      strcat(datafile, ".shf" );

      if(  ( fptr = fopen( datafile, "a") )== NULL )
        {
         STP( 1, 25);
         printf("Couldn't open datafile!");
         exit(0);
        }
      fprintf(fptr," %s  %d   %s\n", datafile, res_fac, ptr );

      STP(  10, 10 );
      strcpy(txt,"                                        ");
      printf("%s",txt);
      STP(74,25);
      strcpy(txt,"RECORD");
      printf("%s",txt);
     }

   else                     /* don't store anymore */
     {
      STP(74,25);
      strcpy(txt,"        ");
      printf("%s",txt);
      fcloseall();
     }
  }


/*---------- c:  enter comment into velocity datafile -----------------------*/
if(key =='c' && store_flag == 1 )
 {
  STP(3,5);
  sprintf(txt, "Enter Comment:");
  printf("%s", txt );
  STP(3,6);
  gets(txt);
  fprintf(fptr,"\n%s\n", txt);
  strcpy(txt,"              ");
  STP(3,5);
  strcpy(txt,"                                                                           ");
  printf("%s",txt);
  STP(3,6);
  printf("%s", txt);
 }


/*--------------- p:  increment effective PR period --------------------------*/
if( key == 'p' )
  {
   effective_pr_period++;
   if(effective_pr_period > 10 ) effective_pr_period = 1;
   outport(CONTROL, 0);
   outport( WRITE_NUM_ECHOES, 255 );
   outport(CONTROL, 5);
  }


/*--------------- P:  decrement effective PR period --------------------------*/
if( key == 'P' )
  {
   effective_pr_period--;
   if(effective_pr_period < 1 ) effective_pr_period = 10;
   outport(CONTROL, 0);
   outport( WRITE_NUM_ECHOES, 255 );
   outport(CONTROL, 5);
  }
```

```
/*--------------- d:  increment max delta ------------------------------------*/
if( key == 'd' )
  {
   max_delta++;
   if(max_delta > 10 ) max_delta = 1;
  }


/*--------------- D:  decrement max delta ------------------------------------*/
if( key == 'D' )
  {
   max_delta--;
   if(max_delta < 1 ) max_delta = 10;
  }

 /*--------------- s:  increment sets to average ------------------------------*/
if( key == 's' )
  {
   set_av++;
   if(set_av > 24 ) set_av = 1;
   number_echoes = set_av * 10 + 15;
   outport(CONTROL, 0 );
   outport( WRITE_NUM_ECHOES, number_echoes );
   outport(CONTROL,15);
  }

 /*--------------- S:  decrement sets to average ------------------------------*/
if( key == 'S' )
  {
   set_av--;
   if(set_av < 1 ) set_av = 24;
   number_echoes = set_av * 10 + 15;
   outport(CONTROL, 0 );
   outport( WRITE_NUM_ECHOES, number_echoes );
   outport(CONTROL,15);
  }

/*--------------- A:  preset parameter set #1 --------------------------------*/
if( key == 'A' )
  {
    max_delta = 1 ;
    set_av    = 20;
    number_echoes = set_av * 10 + 15;
    outport(CONTROL, 0 );
    outport( WRITE_NUM_ECHOES, number_echoes );
    outport(CONTROL,15);
   }

/*--------------- B:  preset parameter set #2 --------------------------------*/
if( key == 'B' )
  {
    max_delta = 10 ;
    set_av    = 1;
    number_echoes = set_av * 10 + 15;
    outport(CONTROL, 0 );
    outport( WRITE_NUM_ECHOES, number_echoes );
    outport(CONTROL,15);
   }

/*--------------- C:  preset parameter set #3 --------------------------------*/
if( key == 'C' )
  {
    max_delta = 10;
    set_av    = 3;
    number_echoes = set_av * 10 + 15;
    outport(CONTROL, 0 );
    outport( WRITE_NUM_ECHOES, number_echoes );
    outport(CONTROL,15);
   }
```

```
            if(key == 'q') { fcloseall(); return; }

            STP(1,25);
            strcpy(txt," ");
            printf("%s",txt);
            key = '0';

           } /*========================== velocity plotting options =========================================*/


         /*--------------------- plot the flow ---------------------------------------------------------*/
         setcolor(YELLOW);
         y_pt = (int)(350. - vel_scale * yscale * flow [c]);       /* calculate velocity from shifts     */
         moveto( 0, y_pt );
         old_flow[0] = y_pt;

         for(c=1; c<=xmax; c++)                                   /* Plot the flow values               */
             {
              y_pt = (int)(350. - vel_scale * yscale * flow [c]);
              if( y_pt > 639 ) y_pt = 479;                         /* Clip points beyond current scale  */
              if( y_pt < 0   ) y_pt = 0;
              lineto( (int)( (float)c * xscale ), y_pt );
              old_flow[c] = y_pt;
             }



        /*---------------- save the velocity plot? ---------------------------------*/
        if( sav_flag==1 )
          {
           STP(1,25);
           sprintf(txt,"Hit s to save RF for this plot                       ");
           printf("%s",txt);
          }

        if(sav_flag==1 || load_flag ==1 )
          {
           while ( !kbhit() );
           key = getch();
          }
        if(sav_flag == 1 && key == 's' )save_echoes();
        if(key=='q')return;

   }
   /*++++++++++++++++++++++++ end loop until q key hit +++++++++++++++++++++++++++++++++++++++++*/

}
/****************************** end plot flow *******************************************************/
```

```
/*********************************** plot_echo ****************************************************/
/*                                                                                                 */
/*   Displays RF echoes from correlator memory to the screen                                       */
/*                                                                                                 */
/*                                                                                                 */
/***************************************************************************************************/

void plot_echo()
{
  CLS;
  initgraph(&g_driver, &g_mode, "");

/*................................ loop until key hit .....................................*/
while(1)
   {

    /* ---------------------- load start address ---------------------------------------*/
    while( ( value = inport(READ_ADDRESS_COUNTER) ) != 0 )

          {
           outport(CONTROL, 0 );                               /* Load start address, zero */
           outport(LOAD_START_ADDRESS, 0);                   /* Wait until loaded       */
           STP(55,18);
           sprintf(txt,"START Address = %4x",value );
           printf("%s",txt);
           STP(55,19);
           strcpy(txt,"END Address   = waiting" );
           printf("%s",txt);
           if( kbhit() ) { getch(); return; }
          }

           STP(55,19);
           strcpy(txt,"END   Address = dtizing" );
           printf("%s",txt);

           STP(55,18);
           sprintf(txt,"START Address = %4x",value );
           printf("%s", txt);


           /*----------------- acquire data ------------------------------------------------*/
           outport(CLEAR_ND_LATCH, 0 );                            /* clear new data latch */
           outport(COMP_ACQUIRE_DATA, 0 );                          /* acquire data         */

           while(  ( inport( RD_TRANS_POS ) & 0x0080 ) == 1 )      /* wait for data to be  */
                {                                                   /* acquired             */
                 if( kbhit() ) { getch(); return; }
                }
           STP(55,17);
           strcpy(txt,"ACQUIRING DATA");
           printf("%s", txt);

           while(  ( inport( RD_TRANS_POS ) & 0x0080 ) == 0 )
                {
                 if( kbhit() ){ getch(); return; }
                }

           if( kbhit() ) { getch(); return; }

           /*----------------- print various info to screen ------------------------------*/
           value = inport( READ_ADDRESS_COUNTER );
           STP(55,19);

           sprintf(txt, "END   Address = %4x   ",value );       /* print end address value */
           printf("%s",txt);

           STP(55,17);
           strcpy(txt, "              ");
           printf("%s",txt);

           if( sav_flag == 1 ) save_echoes(  );                    /* Want to save RF echoes? */
```

```
outport( CONTROL, 5 );


/*================== main echo plotting loop =============================================*/

for(msa = 0; msa <  display_echoes; msa = msa + 1024 )      /* Plots 1k echoes to screen */
   {

    if( kbhit() ) { getch(); return; }

    /*------------------------ get a 1K echo from memory -------------------------------*/
    for( c=0; c<1024; c++)
       {

        if(display_echoes==1024) msa=1024;
        address = msa+(long)c;

        outport( CORR_ADD_REG_1_LO, address & 0x00000fff );
        outport( CORR_ADD_REG_2_LO, address & 0x00000fff );
        top8 = (int)( address / 4096 );
        outport( CORR_ADD_REGS_HI,  top8 + 256 * top8 );

        if( bank == -1 )
           value = inport( CORR_READ_MEM ) & 0x00FF ;
        else
           {
            value = inport( CORR_READ_MEM ) & 0xFF00 ;
            value = value >> 8;
           }
        values [c] = value;
       }


    /*------------------------ print various information --------------------------------*/
    STP(55,24);
    sprintf(txt, "Add  = %7lx", msa);                          /* start address of echo  */
    printf("%s", txt);
    STP(55,25);
    sprintf(txt, "Echo = %5u", msa/1024 );                     /* echo number            */
    printf("%s", txt);
                                                              /* print out values of the */
                                                              /* the echo at locations 8 */
                                                              /* 9, 10 and 11            */
    STP(55,23);
    sprintf(txt,"%3u %3u %3u %3u", values[8], values[9], values[10], values[11] );
    printf("%s", txt);



   /*----------------- plot the actual 1K echo ------------------------------------------*/
    xoffset = 0;
    yoffset = 0;
    setcolor(14);
    moveto( 0, values[0] + yoffset);

    for( c=1; c<1024; c++)                           /* Plot echo.  Since VGA screen is  */
       {                                             /* only 640x480 and the echo length */
        if( c==640 )                                 /* is 1024, break echo up and put   */
          {                                          /* part on top of screen and rest   */
           xoffset = 640;                            /* near the bottom                  */
           yoffset = 224;
           moveto(0, values[640] + yoffset);
          }
        else
       lineto( c-xoffset, values [c] + yoffset);
      }
```

```
              /*---------------- erase the previous trace ----------------------------------------*/
               xoffset = 0;                                        /* Erase previous trace              */
               yoffset = 0;

               setcolor(0);
               moveto( 0, values[0] + yoffset);

               for( c=1; c<1024; c++)
                  {
                   if( c==640 )
                     {
                      xoffset = 640;
                      yoffset = 224;
                      moveto(0, values[640] + yoffset);
                     }
                   else
                   lineto( c-xoffset, values [c] + yoffset);
                  }

              } /*========== main echo plotting loop ==============================*/

   } /*........................... loop until key hit ...............................*/
}

/****************************** end plot_echo *******************************************************/
```

```
/*********************************** save_echoes ****************************************************/
/*                                                                                                  */
/*     Saves 1K echoes to the file rl-time.ech                                                      */
/*                                                                                                  */
/****************************************************************************************************/

void save_echoes( )

{
  int
       e,
       echo_n              /* echo number */
       ;

  long
      base_add
      ;

  FILE *fecho
       ;

  if( ( fecho = fopen( "rl-time.ech", "w" ) ) == NULL )
    {
     puts("Can't open rl-time.ech!\n");
     exit(0);
    }


       for(echo_n = 0; echo_n < num_rf_ech; echo_n++)
            {
               outport(CONTROL, 5);
               base_add = 1024 * (long)echo_n;
               printf("%d ",echo_n);
               fprintf( fecho, "%d\n", echo_n);
               for( c=0; c<1024; c++)                    /* Read in 1K echo */
                  {

                   address = base_add+(long)c;

                   outport( CORR_ADD_REG_1_LO, address & 0x00000fff );
                   outport( CORR_ADD_REG_2_LO, address & 0x00000fff );
                   top8 = (int)( address / 4096 );
                   outport( CORR_ADD_REGS_HI,  top8 + 256 * top8 );

                   if( bank == -1 )
                      value = inport( CORR_READ_MEM ) & 0x00FF ;
                   else
                      {
                       value = inport( CORR_READ_MEM ) & 0xFF00 ;
                       value = value >> 8;
                      }
                  fprintf( fecho, "%d\n", value );          /* Write to disk     */
                  }
            }
fcloseall();
printf("\n");
}

/****************************** end save_echoes ****************************************************/

/******************************** load_echoes ******************************************************/
/*                                                                                                 */
/*    This routine takes 255 echoes stored on disk and puts them into                             */
/*    the correlator memory. Read in 1K at a time.                                                 */
/*                                                                                                 */
/***************************************************************************************************/

void load_echoes( )

{
  int
```

```
       e,
       echo_1 [1024],       /* 1K echo buffer */
       echo_n = 0           /* echo number    */
       ;

  long
      base_add = 0
      ;

  FILE *fecho
       ;

  if( ( fecho = fopen( "rl-time.ech", "r" ) ) == NULL )
    {
     puts("Can't open rl-time.ech!\n");
     exit(0);
    }

    outport(CONTROL, 15);

    while( echo_n != 255 )
         {
           fscanf( fecho, "%d", &echo_n);                    /* read in echoes          */
           printf("%d ",echo_n);
           base_add = 1024 * (long) echo_n;

           for(c=0; c<1024; c++)                             /* get echoes 1K at a time */
              fscanf(fecho, "%d", &echo_1[c]);

                for(c = 0; c < 256; c++)                     /* put in correlator mem   */
                   {
                    e = echo_1[4*c] + 256 * echo_1 [4*c+1];
                    outport(CORR_DATA_REG_12, e );
                    e = echo_1[4*c+2] + 256 * echo_1 [4*c+3];
                    outport(CORR_DATA_REG_34, e );

                    address = base_add + 4 * (long)c;

                    outport( CORR_ADD_REG_1_LO, address & 0x00000fff );
                    outport( CORR_ADD_REG_2_LO, address & 0x00000fff );
                    top8 = (int)( address / 4096 );
                    outport( CORR_ADD_REGS_HI,  top8 + 256 * top8 );
                    outport(CORR_WRITE_MEM,0);
                   }

         }
}
/****************************** end load echoes *****************************************************/
```

```
/********************************* correlate ***************************************/
/*                                                                                */
/*                     routine that controls correlator                           */
/*                                                                                */
/*                          written by Jerome Chen                                */
/*                                                                                */
/**********************************************************************************/

#include        <stdio.h>
#define DELAY           5
#define THRESH1         10250
#define THRESH2         55286
#define MAX_ERRORS      5

int errors = 0;

init()
{
        outport(0x310, 0x45);
        outport(0x30e, 0);
        outport(0x30e, 0);
}

correlate(start1, start2)
long start1, start2;
{
        int i;
        unsigned int result;

        errors = 0;

redo:
        outport(0x300, start1);
        outport(0x302, start2);
        outport(0x304, (((start1 >> 12) << 8) | (start2 >> 12)));
        outport(0x30e, 0);

        for(i = 0; i < DELAY; i++);

        result = (unsigned int)inport(0x30c);

        if(result > 28791)
                result -= 57583;

        if(start1 == start2)
        {
                if(result > 32767)
                {
                        if(errors++ > MAX_ERRORS)
                                return(1);
                        goto redo;
                }
                else if(result == 0)
                        result++;
        }
        else if((result > THRESH1) && (result < THRESH2))
        {
                if(errors++ > MAX_ERRORS)
                        return(THRESH1);
                goto redo;
        }

        return(result);
}
/********************************* end correlate.c ******************************************/
```

```
/****************************************** do_corr6 *********************************/
/*                                                                                    */
/*                                                                                    */
/*      Correlation algorithm.  Written by Jerome Chen, based on the temporary        */
/*      system program      (Appendix E)                                              */
/*                                                                                    */
/**************************************************************************************/

#include <math.h>

#define DELTA_MAX       10
#define COR_LENGTH      40
#define MIN_SHIFT       2
#define MAX_SHIFT       7
#define ECHO_LENGTH     1024


#define SQRT(x)                 (float)sqrt((double)(x))
#define FABS(x)                 (float)fabs((double)(x))


extern int base_echo,
           max_delta,
           effective_pr_period,
           effective_pr_period
           ;

extern char key;

int be;
int range;
long base1;
long base2;

int b;
int i;
extern int c;
int cor_ctr;
int delta;
int offset;
int shift;
int start_index[11];
int max_range;
int res_inc;

float sum_e1_x_e2;
float sum2_e1;
float sum2_e2;

float tmp;
float guess;
float peak;
float correlation[11];
float peak_sum[11];
float peak_sum_2[11];
float tshift[11];
float tshift_2[11];

float shvar;
float den;
float mean;
extern float flow[];


corr(sets, disp)
int sets, disp;
{
        b = 0;
        be = base_echo;
        if      ( disp == 1 ) { res_inc = 40;  max_range = 900; }
        else if ( disp == 2 ) { res_inc = 20;  max_range = 450; }
        else if ( disp == 3 ) { res_inc = 10;  max_range = 225; }



        init();
        for(i = 0; i < 100; i++)
                flow[i] = 0.0;

        for(i = 0; i < sets; i++)
        {
         if( kbhit() )
           {
            key = getch();
            if( key == 'q' ) return;
           }


                for(range = 10; range <= max_range; range += res_inc)
                {
                        corr_20();

                        mean = 0.0;
                        den = 0.0;

                        for(c = 1; c <= max_delta; c++)
                        {
                                tshift[c] = tshift[c] / (float)c;
                                tshift_2[c] = tshift_2[c] / (float)c;

                                shvar = tshift_2[c] - tshift[c] * tshift[c];

                                if(shvar <= 0.0)
                                        shvar = 10.0 * fabs(tshift[c]);

                                if(fabs(tshift[c]) < .01)
                                        tshift[c] = .01;


                                den += 1.0 / shvar;
                                mean += tshift[c] / shvar;
                        }

                        flow[b++] += mean / den;
                }

                b = 0;
                be += 10;
        }

        if(sets > 1)
                for(i = 0; i < 100; i++)
                        flow[i] /= (float)sets;
}



corr_20()
{

        for(c = 0; c <= DELTA_MAX; c++)
        {
                peak_sum[c] = 0;
                start_index[c] = 0;
                tshift[c] = 0;
                tshift_2[c] = 0;
        }
```

```
    for(delta = 1; delta <= max_delta; delta++)
    {
        guess =  tshift[delta - 1] *
            (((float)delta + 1.0) / (float)delta);


        for(offset = 0; offset < max_delta; offset++)
        {

            base1 = ((long)(be + offset) * ECHO_LENGTH) +
                start_index[offset] + range;
            base2 = base1 + effective_pr_period * (delta * ECHO_LENGTH);

            shift = (int)guess - MIN_SHIFT;
            cor_ctr = 0;

            sum2_e1 = (float)correlate(base1, base1);

            while((sum_e1_x_e2 =
                (float)correlate(base1, base2 + (long)shift)) < 0.0
                && cor_ctr < 3 )
            {
                shift++;
                cor_ctr++;
            }

            if(cor_ctr < 6)
            {
                sum2_e2 = (float)
                    correlate(base2 + (long)shift, base2 + (long)shift);

                correlation[cor_ctr] = sum_e1_x_e2 /
                    SQRT(sum2_e1 * sum2_e2);

                do_correlation();

                while(cor_ctr < MAX_SHIFT)
                {
                    do_correlation();

                    if(correlation[cor_ctr] <
                        correlation[cor_ctr - 1])
                        break;
                }

                find_peak();
            }
            else
                peak = (float)(shift - 4);

            peak_sum[delta] += peak;
            peak_sum_2[delta] += peak * peak;

            if(delta == 1)
                start_index[offset + 1] = (int)peak_sum[delta];
        }

        tshift[delta] = peak_sum[delta] / max_delta;
        tshift_2[delta] = peak_sum_2[delta] / max_delta;
    }
}
```

```
do_correlation()
{
        shift++;
        cor_ctr++;

        sum_e1_x_e2 = (float)correlate(base1, base2 + (long)shift);
        sum2_e2 = (float)correlate(base2 + (long)shift, base2 + (long)shift);

        correlation[cor_ctr] = sum_e1_x_e2 / SQRT(sum2_e1 * sum2_e2);
}


find_peak()
{
        if(correlation[cor_ctr] < 0 )
                peak = (float)(shift - 1);
        else if(correlation[cor_ctr - 1] < correlation[cor_ctr - 2])
                peak = (float)(shift - 2);
        else
        {
                tmp = 2 * (correlation[cor_ctr - 2] - 2 *
                        correlation[cor_ctr - 1] + correlation[cor_ctr]);

                if(FABS(tmp) < .001)
                        peak = (float)(shift - 1);
                else
                        peak = (float)shift - 1 + (correlation[cor_ctr - 2] -
                                correlation[cor_ctr]) / tmp;
        }

        if(peak < -20.0 || peak > 63.0)
                peak = (float)(shift - 1);
}

/********************************************** end do_corr6 *******************************************/

^Z
```

VITA

Ilmar Arthur Hein was born August 15, 1959, in Woodstock, Illinois. From 1977 to 1981 he attended the University of Illinois at Urbana-Champaign as an undergraduate and was awarded a Bachelor of Science degree in Electrical Engineering.

From 1981 to 1983 Mr. Hein was enrolled in the Master of Science program in Electrical Engineering at the University of Illinois. His interests included electromagnetics; he also worked in collaboration with a circadian rhythm research laboratory at the Department of Psychology. He was awarded a Master of Science degree in 1983 and produced a thesis entitled "A Computerized Data Collection and Control System for Physiological Research Laboratories."

From 1983 to 1985, Mr. Hein was employed by Hughes Aircraft Company, Space and Communications Division, in El Segundo, California, as a microwave engineer.

From 1985 to 1990, Mr. Hein worked in the Bioacoustics Research Laboratory of the University of Illinois as a research assistant on his doctoral dissertation and other ultrasonic and bioengineering-related research. In 1988 he was awarded the Terrence Matzuk Award from the American Institute of Ultrasound in Medicine for innovative research in the development of ultrasonic instrumentation and technology. He was also a teaching assistant for a number of electrical engineering courses. He was voted an excellent teaching assistant by his students and was nominated for the Harold L. Oleson Award for Excellence in Undergraduate Teaching by Graduate Students.