Development of the In Vivo Flow Cytometer

by

John P. Novak

S.B. Aeronautics/Astronautics
MIT 1980
S.B. Mechanical Engineering
MIT 1980
S.M. Mechanical Engineering
MIT 1995
S.M. Nuclear Engineering
MIT 1995

SUBMITTED TO THE DEPARTMENT OF MECHANICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTORATE OF PHILOSOPHY IN MECHANICAL ENGINEERING
AT THE
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SEPTEMBER 2004

Signature of Author: _____
Department of Mechanical Engineering
July 16, 2004

Certified by: _____
Charles Lin
Assistant Professor of Biophysics
Thesis Supervisor

Accepted by: _____
Ain Sonin
Chairman of Graduate Studies

_____
C. Forbes Dewey
Professor of Mechanical Engineering

_____
Roger Kamm
Professor of Mechanical Engineering, Bioengineering

DEVELOPMENT OF THE IN VIVO FLOW CYTOMETER

By

JOHN NOVAK

Submitted to the Department of Mechanical Engineering
on July 16, 2004 in partial fulfillment of the
requirements for the Degree of Doctorate of Philosophy in
Mechanical Engineering

ABSTRACT

An in vivo flow cytometer has been developed that allows the real-time detection and quantification of circulating cells containing fluorescent proteins or labeled with fluorochrome molecules in live animals, without the need to extract blood samples. A stationary laser beam is focused by a cylindrical lens to a slit of light that is then demagnified and focused across a blood vessel by an achromat and microscope objective. Fluorescent cells are excited one by one as they flow through the excitation laser light slit, creating a burst of fluorescence whose width is inversely proportional to their velocity. The fluorescence signal is detected through a confocal slit aperture using a photomultiplier tube. The analog signal from the photomultiplier tube is then digitized, filtered, and recorded as a function of time onto a computer. Computer programs post-process the data for the presence of cell signal, as well as various aspects of the cell signal such as height, width, and temporal location of the signal peak.

Two in vivo flow cytometers have been built: a single-slit, single-color system and a two-slit, two-color system. The single-slit, single-color system provides excitation at 632 nm, and the two-slit, two-color system provides excitation at 632 nm and 473 nm. The two-slit, two-color system can operate in several different modes: single-slit at 632 nm or 473 nm, double-slit at 632 nm or 473 nm, and double-slit with one excitation slit at 632 nm and the other at 473 nm. Thus far, the single-slit, single-color system has been used to study the circulation kinetics of different prostate cancer and leukemia cell lines with different metastatic potential, as well as the effect of different host environments (i.e., mouse versus rat). In addition, the device has been used to develop a new in vivo labeling method of white blood cells that does not result in significant depletion of the labeled cells, allowing for the possibility of autoimmune and transplant rejection studies. The two-slit, two-color system is being used to track two different cell populations, or one cell population labeled with two different markers, one of which can be the green fluorescent protein.

Thesis Advisor: Charles Lin

Title: Assistant Professor of Biophysics

Table of Contents

Chapter 1

Introduction: Research Problem Definition and the Proposed Solution

The study of tumor cells circulating in the bloodstream and their ability to form secondary tumors has been an area of research focus for numerous years. The reason for this is that death due to such cancers as breast, prostate, and colon is primarily caused by secondary tumors that metastasize from the primary tumor. However, despite the advances made in malignant tumor (i.e. cancer) research over the last several decades, many questions still remain to be answered about this medical malady. For example, it is hypothesized that the shedding of tumor cells into the circulatory system is one of the key steps in cancer metastasis, but it is not known at what stage or stages in tumor growth that this occurs[1]. It is also not known if the circulating tumor cell (CTC) count is representative of metastatic potential of the tumor, and if the CTC count is indicative of tumor burden[1]. In addition, it remains to be discovered what correlation exists between CTC count and the patient's (human as well as animal) response to malignant tumor therapy (such as hyperthermia, ultrasound, and photodynamic treatment), and whether malignant tumor therapy may actually initiate or accelerate the metastasis process via an increase in the CTC count.

One of the current methods for the detection and counting of circulating tumor cells is (ex vivo) flow cytometry. This method involves the extraction of blood cells from the patient or test animal, the fluorescent labeling of specific cell populations, and the insertion of the blood cells into a flow cytometer. The standard flow cytometer, comprised of a light source (usually a laser), optics, light filters, light detectors, fluid lines, and electronics, passes the cells of the withdrawn blood sample in a single file through the light source and determines, via analysis of the fluorescent signal and the forward and orthogonally scattered light, the types and number of cells present.

Another contemporary device employed to detect and count tumor cells is a hemocytometer. The hemocytometer is a cell-counting chamber. It is comprised of a grid composed of small squares, a cover glass mounting support, and a cover glass. The cover glass is placed on top of the cover glass mounting support to create two counting

chambers. A solution created from a withdrawn blood sample and containing labeled tumor cells is introduced into the counting chambers using a pipette. The hemocytometer is placed on a microscope stage, and labeled tumor cells are selected and counted visually against the grid, located underneath the counting chambers (i.e., underneath the cover glass mounting support). Based on the number of cells counted, the number of squares of the grid occupied by the cells counted, the depth of the counting chamber, and the volume of the blood sample withdrawn, the number of tumor cells in the blood sample is calculated.

Both of these methods do have merit in that both are sensitive and specific. However, a distinct disadvantage to both techniques is that they are invasive in that extraction of blood is required. In addition, each extraction of blood provides only a single time data point. Thus, obtaining a valid temporal population profile for an unknown result or characterization of a process involving rapid changes in the number of circulating cells is problematic, and would require numerous blood extractions closely spaced in time. For test animals such as mice and rats, which have a limited blood volume, such biological studies would require the usage of many animals to avoid affecting their physiology (and, hence, invalidating the test results) due to excessive blood loss. However, the usage of numerous animals would then raise questions concerning host variability, again bringing into question the validity of the results. In addition, for numerous blood extractions as well as numerous hosts, and for the time delays involved between blood extraction and blood analysis, there is ample opportunity for sample contamination as well as damage to the cells of interest, resulting in erroneous data[2].

Non-invasive methods such as positron emission tomography[3], high-resolution magnetic resonance imaging[4], intravital microscopy[5], confocal imaging[6], bioluminescence[7], and two-photon imaging[8] do exist, and have been used to visualize and study the different steps of tumor progression and tumor metastasis. However, even though these techniques span a wide range of spatio-temporal resolutions, none of them have been optimized for detecting quantitative changes in the number of circulating tumor cells. For example, confocal imaging can provide visual images of fluorescently labeled tumor cells flowing through a blood vessel, but counting usually must be

performed manually. For high cell velocities and/or large numbers of labeled cells, accurate cell counting is virtually impossible. The images can be viewed in slow motion after the acquisition of the data, but it can still be difficult (and fatiguing) to discern fast-moving individual cells or cells in a group. In addition, post-processing of the data requires that the images be stored during the test, resulting in extremely large computer files. Software does exist for post-processing analysis, but is slow, difficult to use, and of limited accuracy. Finally, the acquisition of good images requires the usage of a high numerical aperture lens, limiting the depth of focus of the system and, hence, affording the possibility, except for very small diameter blood vessels, that labeled cells will pass by out of the depth of focus of the system.

To remedy these problems, an in vivo flow cytometer has been developed with the capability to detect and count circulating fluorescently labeled cells in live, adult, fully-furred animals, without the need to extract blood samples. In addition, this technique allows for the continuous monitoring of the labeled cells of interest in one animal, and it yields quantitative information on the labeled cells without affecting the physiology of the host animal. The underlying principle of operation of the in vivo flow cytometer is confocal excitation and detection of the fluorescently labeled cells in circulation. Light from a laser is focused into a slit by a cylindrical lens and then demagnified and imaged across a blood vessel using an achromat lens and a microscope objective (see Figure 1.1). The blood vessel is chosen using another aspect of the cytometer called the transillumination section. The slit of light across the blood vessel excites fluorescent proteins within or fluorescent labels that have been attached to the circulating cells in the blood stream of which one would like to detect. The fluorescence that results when the fluorescent protein or the fluorochrome molecules are excited by the slit of laser light is gathered by the microscope objective and directed, using mirrors and dichroic beam splitters, to a mechanical slit which is confocal with the slit of laser light imaged across the blood vessel. This confocalness of the mechanical slit with the slit of laser light focused in the blood vessel provides blockage of light from sources not in the blood vessel. A photomultiplier tube directly behind the mechanical slit provides detection of the confocal fluorescence from the circulating labeled cells in the blood

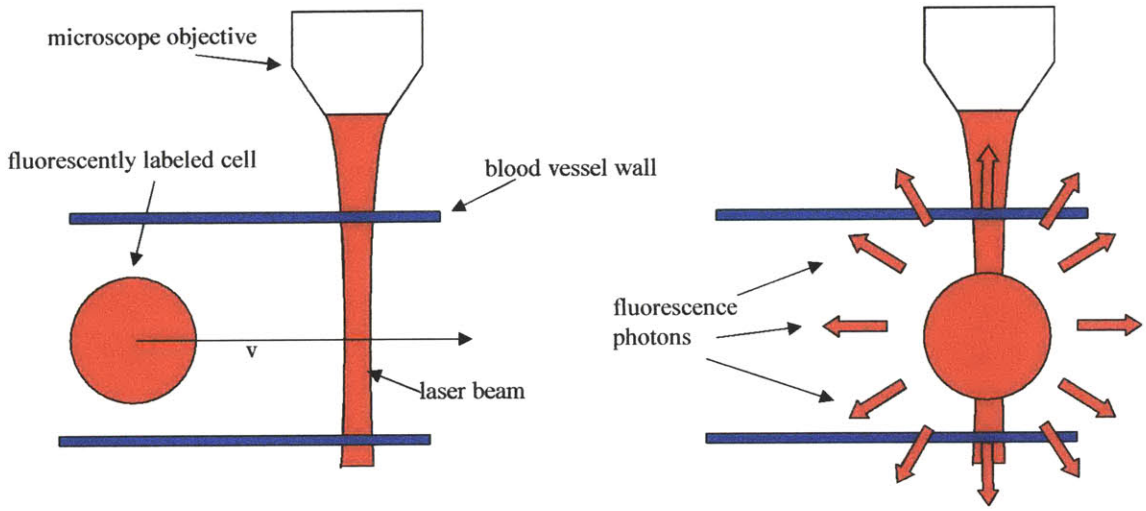

Figure 1.1 Excitation of a labeled cell by laser light slit

vessel. The analog signal from the photomultiplier tube is digitized and stored on a computer. Computer codes analyze the data for cell signal.

Fluorescent labeling of the cells to be detected with fluorochrome molecules is accomplished by intravenous injection of the fluorochrome molecules conjugated to antibodies that specifically bind to the target cells. Alternatively, cells are labeled ex vivo and introduced into animals in adoptive transfer experiments. Fluorescent sources on the cells of interest can also be created by the transvection of a gene into the cells, before the start of the experiment, which creates a fluorescent protein in them (such as GFP).

Thus, this method allows for continuous acquisition of data from one animal with no need of blood extraction, eliminating the possibility of altering the physiology of the test animal due to too much blood loss, or the need to have numerous test animals involved in the experiment. In addition, the possibility of erroneous data due to blood sample contamination or blood sample damage is completely eliminated. This technique also provides test results with minimal time delay, and no special preparation of the animal (such as the shaving of the fur of the animal) is required for data acquisition. Although this method allows for data acquisition on the cells of interest only through fluorescent tagging (since forward and orthogonally scattered light will not be able to be collected), this should not prove to be too heavy of a penalty, since the information desired can be obtained by analyzing the fluorescent signals recorded. Indeed, this

technique has already been used to examine the circulation kinetics of different prostate cancer and leukemia cell lines with different metastatic potential, as well as the affect of different host environments (i.e., mouse versus rat). In addition, the device has been used to develop a new labeling method of white blood cells that does not result in significant depletion of the labeled cells, allowing for the possibility for autoimmune and transplant rejection studies. In short, this method allows for acquisition of data (from mice and rats) which could be used to answer or help to answer many biological questions, including the above questions concerning time course of tumor cell shedding, metastatic potential of the tumor, tumor burden, and degree of correlation of CTC count with animal responsivity to tumor therapy. If successful, this process could then be applied to human subjects in the future, although it is hoped that the data obtained from animal studies can be applied to humans as well.

In what follows, Chapter 2 gives a thorough physical description and explanation of the in vivo flow cytometers that have been built, and Chapter 3 discusses the computer codes that have been written to analyze the acquired data for cell signal. Chapter 4 discusses the various applications of the devices that has already occurred, as well as recommendations for changes to improve the in vivo flow cytometer systems and future tests to characterize the systems (i.e., find the detection limits of performance of the devices), along with more biological tests to perform.

Chapter 2

System Description

Introduction

Two in vivo flow cytometers have been designed and built. The first system is a single-slit, single-color system. The excitation wavelength is 632 nm, provided by a Helium-neon laser. This wavelength was chosen first because of its good penetrating capability through tissue and blood, as well as its strong excitation capability of the fluorochromes DiD and PE-Cy5, to be utilized in planned experiments. In addition, this wavelength is sufficiently far from the emission wavelengths of DiD and PE-Cy5 where fluorescent detection will take place, allowing for filtering of reflected excitation light from fluorescence light. The second system is a two-slit, two-color system, with excitation wavelengths of 473 nm and 632 nm. The 473 nm wavelength was selected because of its capability to fluorescently excite cells expressing the EGFP gene. A diode-pumped solid state diode laser provides the shorter wavelength excitation light, and a Helium-neon laser, like the first system, produces the longer wavelength excitation light. The two-slit, two-color option allows for detection of two different labeled cell populations, as well as detection of cells possessing two different fluorescent sources. This second system can also operate as a two-slit, single color system, using either laser as the source for excitation. Operation of the system in this mode allows for accurate determination of cell velocity, since the distance between the slits and the time difference between the signal peaks can be accurately determined. Also, cell detection accuracy of the device is enhanced in this mode of operation since two voltage waveforms can be compared against one another. An acoustic optical modulator through which the laser beams are directed provides for this option of laser beam diffraction.

Both systems have three aspects to their physical construction: the transillumination section, the fluorescent excitation section, and the detection section. The purpose of the transillumintion section is to provide visualization of the vasculature at depths of up to 100 microns on the area of the animal's body from which data is going to be acquired. This allows for selection of an appropriate blood vessel, both in type and size, as well as an appropriate location of data acquisition on the blood vessel, and

repeatability in finding the same data acquisition site. Discriminating between a vein and artery is required since there is a significant difference in the characteristics of the flow in these two types of blood vessels. Venular flow is much slower than arterial flow in general, and some cells will also slowly roll along the wall of a vein (something not found in arterial flow). Selection of a blood vessel of appropriate diameter is needed so that statistically acceptable data can be acquired in a reasonable time period, with minimal occurrence of more than one labeled cells traveling through the probe volume. Calculations indicate an arteriole approximately 30 microns in diameter will meet these requirements. Appropriate location of the data acquisition site is important to avoid overlying structures such a hair, hair follicles, melanin patches, and sebaceous glands, which will attenuate, through absorption and scattering, the excitation energy and the fluorescence energy. Repeatability in finding the same data acquisition site is required for valid temporal studies, since the flow or signal (excitation or fluorescence) at another data acquisition location can be different from the original site due to the changing influence of various physiological or environmental parameters.

The ear of the test animal is the area chosen to transilluminate. This area of the body is chosen because a significant amount of light is transmitted by the nonvasculature tissue of the ear due to its minimal thickness. Methocel 2% is used to adhere the ear of the animal to a microscope slide for transillumination and viewing. This adhesive is used to avoid irritation of the tissue, and to help match refractive indices at the surface of the ear to allow for a higher percentage of transmission of the transilluminating light into the ear tissue.

The fluorescence excitation section produces the slit of light which causes the fluorescence of the labeled cells. This section provides the excitation energy via a laser light and shapes the laser light into the appropriate dimensions using lenses and irises. The dimensions of the excitation slit are chosen such that the width of the excitation slit is less than the diameter of the labeled cells (approximately 8 microns), and the height of the slit is comparable to the diameter of the blood vessel chosen for data acquisition. A slit width of approximately 5 microns is used. A narrow excitation width is required so that excitation of more than one cell at a time is kept to a minimum, and that the slit of light is of sufficient intensity to cause detectable fluorescence after traveling through the

skin to reach the blood vessel. An excitation depth of approximately 115 microns is required, equal to the maximum depth of blood vessel selection plus the radius of the blood vessel. The height of the slit is chosen to match the diameter of the blood vessel so that there is reasonable probability that all labeled cells traveling in the vessel will be excited, and that autofluorescence from nonvasculature tissue, which is a noise source, will be kept to a minimum.

The detection section determines the presence of a labeled cell through the detection of fluorescence photons. Lenses, dichroic beam splitters, and mirrors direct fluorescence photons gathered by a microscope objective through a filter and focus them onto a mechanical slit confocal with the excitation slit provided by the fluorescence excitation section. The photons passing through the slit are amplified by a PMT, which produces a triangular pulse-shaped voltage signal, indicative of a labeled cell present.

The above physical aspects of the in vivo flow cytometers are discussed further below. Included is a description of the specific components comprising each physical section of the single-slit and two-slit flow cytometer, the purpose of the components, and how the components operate together. In addition, results of sensitivity tests for the 473 nm channel of the two-slit, two-color system using fluorescently labeled beads are given and discussed.

Single-slit, Single-color System

The single-slit, single-color system is shown in Figure 2.1. As mentioned in the introduction, it is the first in vivo flow cytometer system that was built, and is comprised of a transillumination section, a fluorescent excitation section, and a detection section.

Transillumination Section

As discussed above, the purpose of the transillumination section is to transilluminate the part of the test animal (the ear) from which data will be acquired to allow for selection of either a vein or an artery, as well as selection of a specific location on the selected blood vessel for data acquisition. The transillumination section of the single-slit, single-color system is shown in Figure 2.2. The transillumination section of the device is comprised of the sample stage, light emitting diode (LED), fiber optic cable,

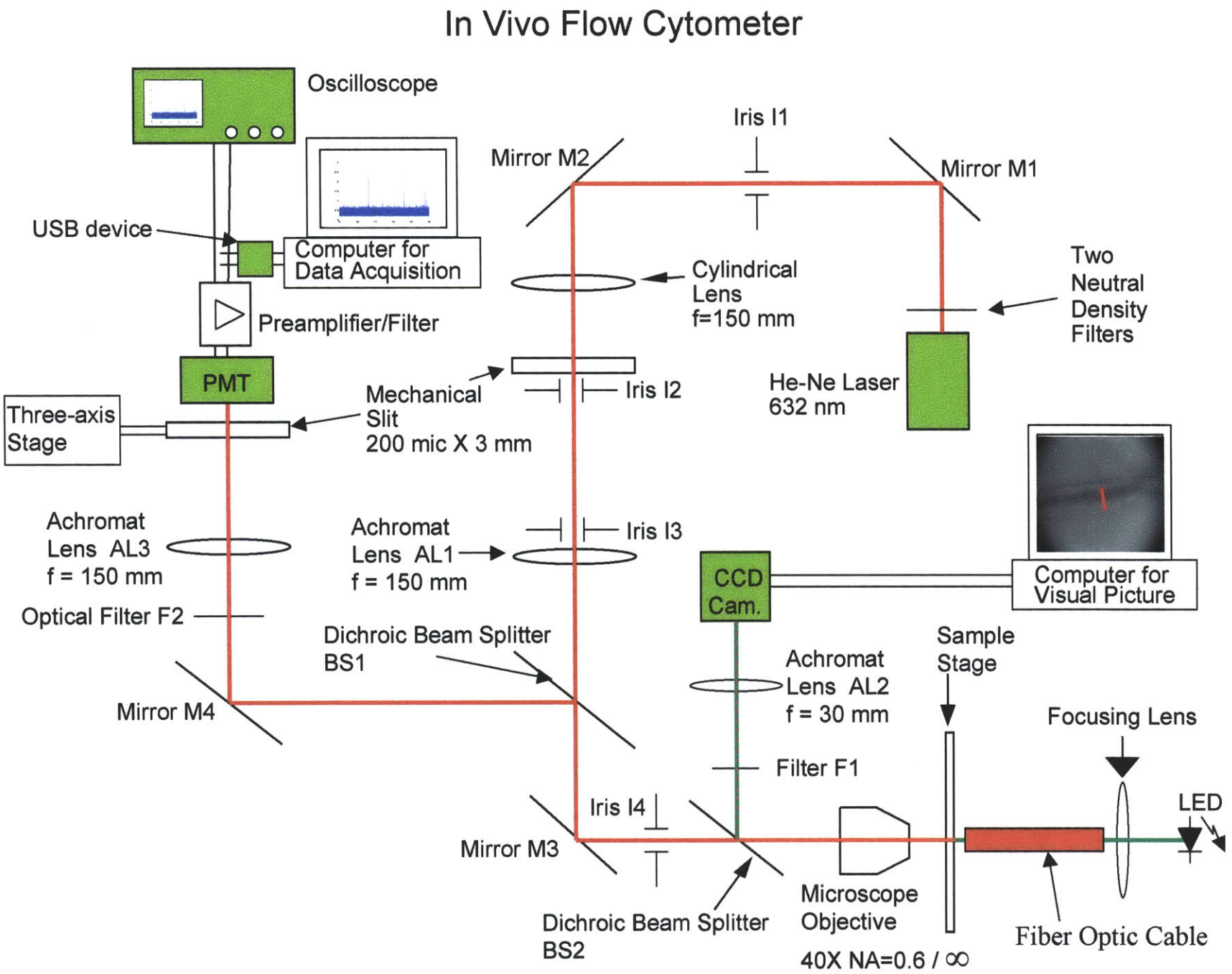

Figure 2.1 The Single-slit, Single-color In Vivo Flow Cytometer

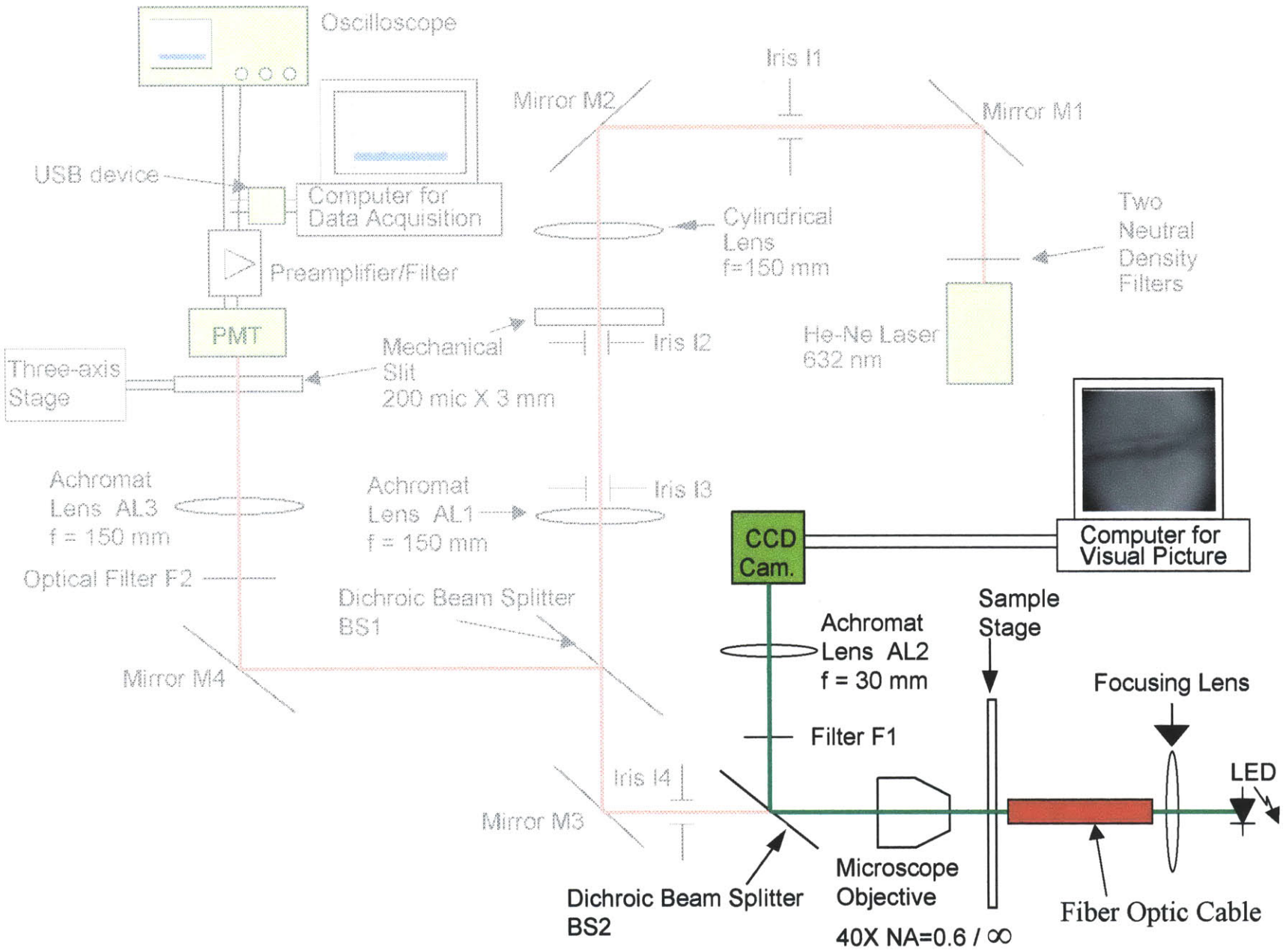

Figure 2.2 The Transillumination and Imaging Subsystem for the Single-slit, Single-color In Vivo Flow Cytometer

microscope objective (Olympus 40X, 0.6 Numerical Aperture, infinity corrected), dichroic beam splitter BS2 (located underneath the sample stage and microscope objective), charged-coupled-device (CCD) camera, achromat lens AL2 (focal length is 30 mm) located in front of the CCD camera, and optical filter F1 (located in front or the achromat lens).

The sample stage is where the test animal from which data will be acquired is situated. The sample stage, comprised of two separate subsystems, provides for temperature and orientation control of the test animal. Temperature control is provided by a heater tape (Minco HR5252R20.3L12A) wrapped around a plastic cylinder (into which the animal is placed) and interfaced with a thermocouple (Physitemp SST-1), a variable DC regulated power supply (Physitemp MW122A), and a temperature controller (Physitemp TCAT2). The thermocouple provides temperature measurement of the animal's body, which is fed into the temperature controller. The temperature controller, where the desired temperature is specified, controls the on/off state of the DC power supply. The DC power supply provides electrical power to the heater tape. Thus, the DC power supply is on causing thermal energy to be created by the heater tape when the temperature measurement from the thermocouple is below the user-specified set temperature of the temperature controller.

Orientation control is provided by a three-axis linear stage, onto which is situated a custom-made rotatable plastic stage. The three-axis stage allows for translational positioning control of the animal, and the rotatable stage allows for angular orientation control of the animal. Both stages have an aperture in their center across which a microscope slide is positioned to which the test animal's ear is adhered. The two stages are interfaced via these apertures, with the rotatable stage having a piece of plexiglass tubing, which is centered around its aperture, that inserts into the aperture of the three-axis stage.

The LED (Future Electronics LXHL-MM1D) provides the transillumination light. The circuit shown in Figure 2.3 adjusts the optical power output of the LED. The circuit functions by maintaining a constant voltage drop across the 100-ohm resistor (i.e., a constant current through the 100-ohm resistor). This constant voltage drop is maintained by the LMS1587-ADJ voltage regulator (NationalSemiconductor). As the variable

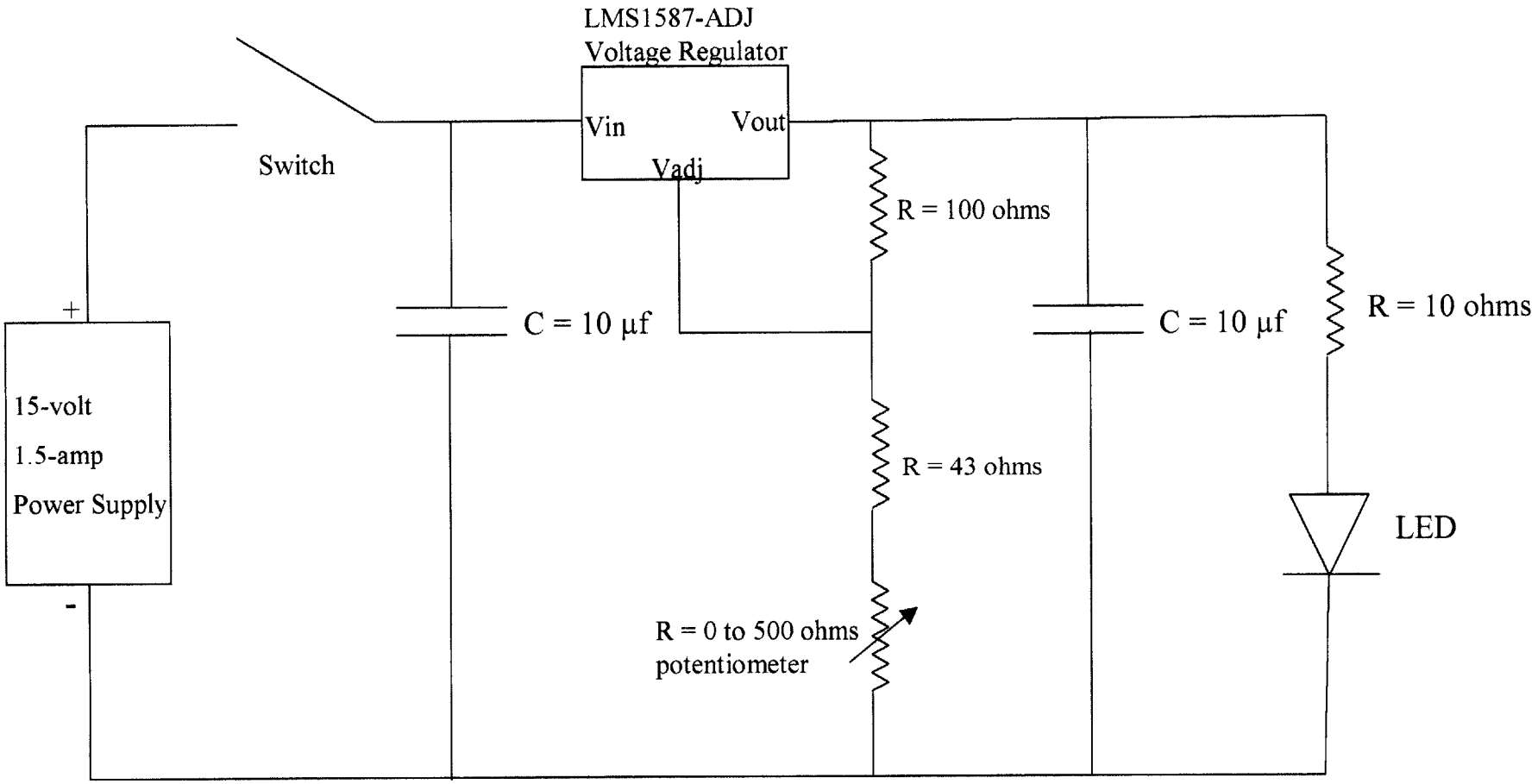

Figure 2.3 Controlling Circuit of LED

resistor (i.e., potentiometer) increases in value, the voltage drop across the entire leg that is comprised of the 100-ohm resistor, the 43-ohm resistor, and the variable resistor (0 to 500 ohms) increases. Consequently, the parallel leg comprised of the 10-ohm resistor and LED has its voltage drop (and, hence, current) increase, resulting in an increase in the power output of the LED. The capacitors in the circuit remove high frequency current oscillations. The LED emits at a wavelength of 524 nm, which provides good contrast between blood vessels and surrounding tissue due to the high absorption of light at this wavelength by hemoglobin. The light from the LED is brought to the animal via a fiber optic cable, which, due to the ease of positioning of the fiber optic tip, allows for good illumination of the ear no matter how the animal is situated on the stage. The transmitted light through the ear enters and passes through the microscope objective, reflects off beam splitter BS2 (Omega Optical) situated below the mounting stage (the beam splitter transmits wavelengths above 600 nm and reflects those below, providing separation of the green LED light from red fluorescence emission), passes through optical filter F1 (Omega Optical) to attenuate its intensity, and is focused onto the CCD camera (Edmund Optics LCL-902C) by the 30 mm focal length achromat lens AL2 (Melles Griot). This achromat lens is located at a distance of 30 mm in front of the CCD camera so that only transilluminating light that has scattered at the focal plane of the microscope objective is in focus. The microscope objective and achromat lens together provide 6.67X magnification. The image from the CCD camera is displayed on a computer screen (CRT), from which an appropriate blood vessel and data acquisition location are chosen. The pixel size of the CRT is 1.47 microns. The field of view of the transilluminating system is measured to be 800 microns X 1000 microns. The resolution of the transillumination system is 0.43 microns.

The Fluorescence Excitation Section

The fluorescent excitation section is shown in Figure 2.4. It is comprised of a He-Ne laser, two neutral density wheels, four irises (I1 through I4), three mirrors (M1 through M3), a cylindrical lens, a mechanical slit, an achromat (AL1), two dichroic beam splitters (BS1 and BS2), and a microscope objective. The Helium-Neon laser provides the energy to fluorescently excite the labeled cell. As mentioned in the introduction, an

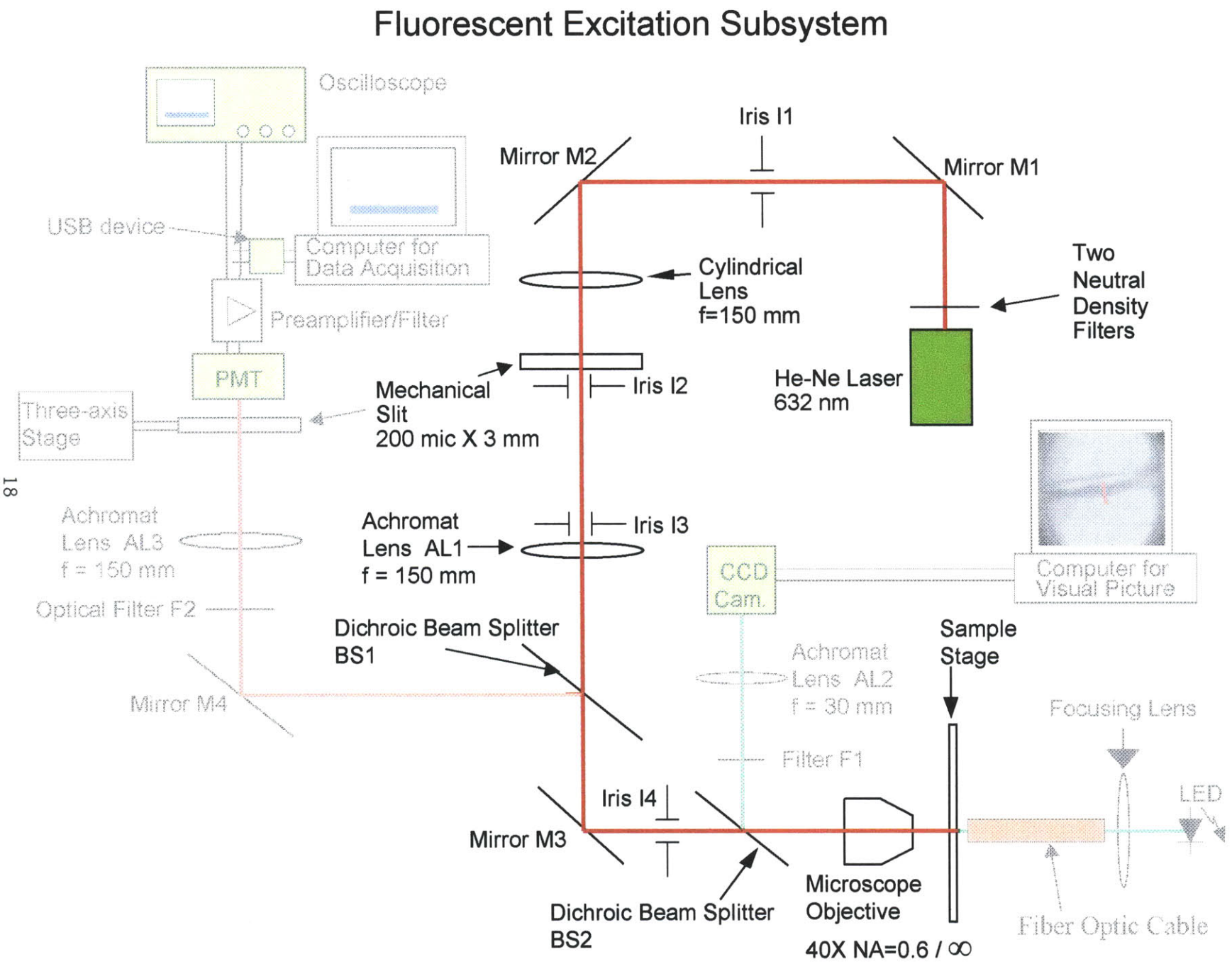

Figure 2.4 Fluorescent Excitation Subsystem for the Single-slit, Single-color In Vivo Cytometer

excitation of 632 nm was chosen for its good penetration through tissue and blood, and its efficient excitation of the fluorochromes DiD and PE-Cy5, to be used in planned experiments. Also, this wavelength is sufficiently far from the emission wavelengths of DiD and PE-Cy5 where fluorescent detection is taking place, allowing for filtering of any reflected excitation light from fluorescence light reaching the detection section. The power entering into the system from the laser is controlled by two neutral density wheels (New Focus Model 5215) situated in front of the laser, with each wheel containing five attenuating filters of different value. The collimated portion of the laser beam is obtained by spatially filtering the laser light through an iris (iris I1). This spatial filtering is required because the Helium-neon laser has a weak spatially incoherent component. Mirror M1 directs the laser light through iris I1. The resulting collimated beam of light is then directed, using mirror M2, into a 150 mm focal length cylindrical lens (Melles Griot), which focuses it into a slit of light onto a mechanical slit 3 mm by 200 microns (Edmund Optics R39-731). The iris immediately after the mechanical slit, iris I2, is used to obtain the desired length of the excitation slit at the sample stage. A 150-mm-focal-length achromat lens (AL1), situated 150 mm beyond where the mechanical slit is located, converts the slit of light into one rotated by 90 degrees and focused at the back focal plane of the microscope objective. (Both dichroics, BS1 and BS2, located in the beam path between the achromat AL1 and the microscope objective, are transmissive at 632 nm. The dichroic BS1 [Omega Optical] is a short pass filter that turns off at approximately 650 nm, transmitting light with a wavelength below 650 nm and reflecting light with a wavelength above 650 nm. The other dichroic, BS2, is a long pass filter that turns on at approximately 600 nm, reflecting light with wavelengths below 600 nm and transmitting light above 600 nm.) The laser excitation light then enters the microscope objective (the same microscope objective of the transillumination section) where it is refocused to the front focal plane of the microscope objective. The width of this slit of light at the front focal plane of the microscope objective is approximately 5 microns. This dimension is much larger than that predicted by diffraction theory and is primarily due to the underfilling of the entrance aperture of the microscope objective in the direction of the length of the slit of light at the back focal plane of the objective. This underfilling is required to provide an excitation slit of uniform intensity over a depth of

approximately 30 microns, the maximum diameter of the blood vessel that would be chosen for data acquisition. The amount of underfilling is determined by the dimension of the excitation laser light horizontal to the optical table at the entrance to the cylindrical lens, the ratio of the focal length of achromat AL1 to the focal length of the cylindrical lens (which is one for this setup), and the back focal length of the microscope objective. The less the objective is filled, the more uniform will be the intensity of laser light with depth into the blood vessel. One could avoid this need of underfilling the entrance aperture of the microscope objective by using an objective of much lower numerical aperture, but this would compromise the number of fluorescence photons captured from the labeled cells. The length of this excitation slit of light is made equal to the diameter of the blood vessel chosen for data acquisition, and is adjusted using iris I2, which adjusts the vertical height of the slit created by the cylindrical lens and, hence, adjusts the width of the slit at the back focal plane of the microscope objective. The larger the vertical height of the slit passing through iris I2, the narrower the slit at the back focal plane of the objective (due to diffraction effects), the more the objective is filled in the width direction of the slit at the back focal plane, the longer the length of the excitation slit at the blood vessel. The position and orientation of the blood vessel from which data is to be acquired is adjusted by translation and rotation of the sample stage so that the data acquisition point is intersected by the laser slit, with the 5 micron dimension in the direction of blood flow, and the longer dimension spanning the diameter of the blood vessel. The power of the He-Ne laser at the sample stage is approximately 600 microwatts.

Detection Section

The detection section is shown in Figure 2.5. It is comprised of the sampling stage, microscope objective, two dichroic beamsplitters (BS1 and BS2), two mirrors (M3 and M4), an optical filter (F2), an achromat lens (AL3), a mechanical slit attached to a three-axis stage, a PMT interfaced with a preamplifier and electronic filter, a USB data acquisition device, a computer (with accompanying data acquisition software), and an oscilloscope. The microscope objective in the detection section is used to gather the fluorescence photons originating at the sample stage. Calculations indicate that for

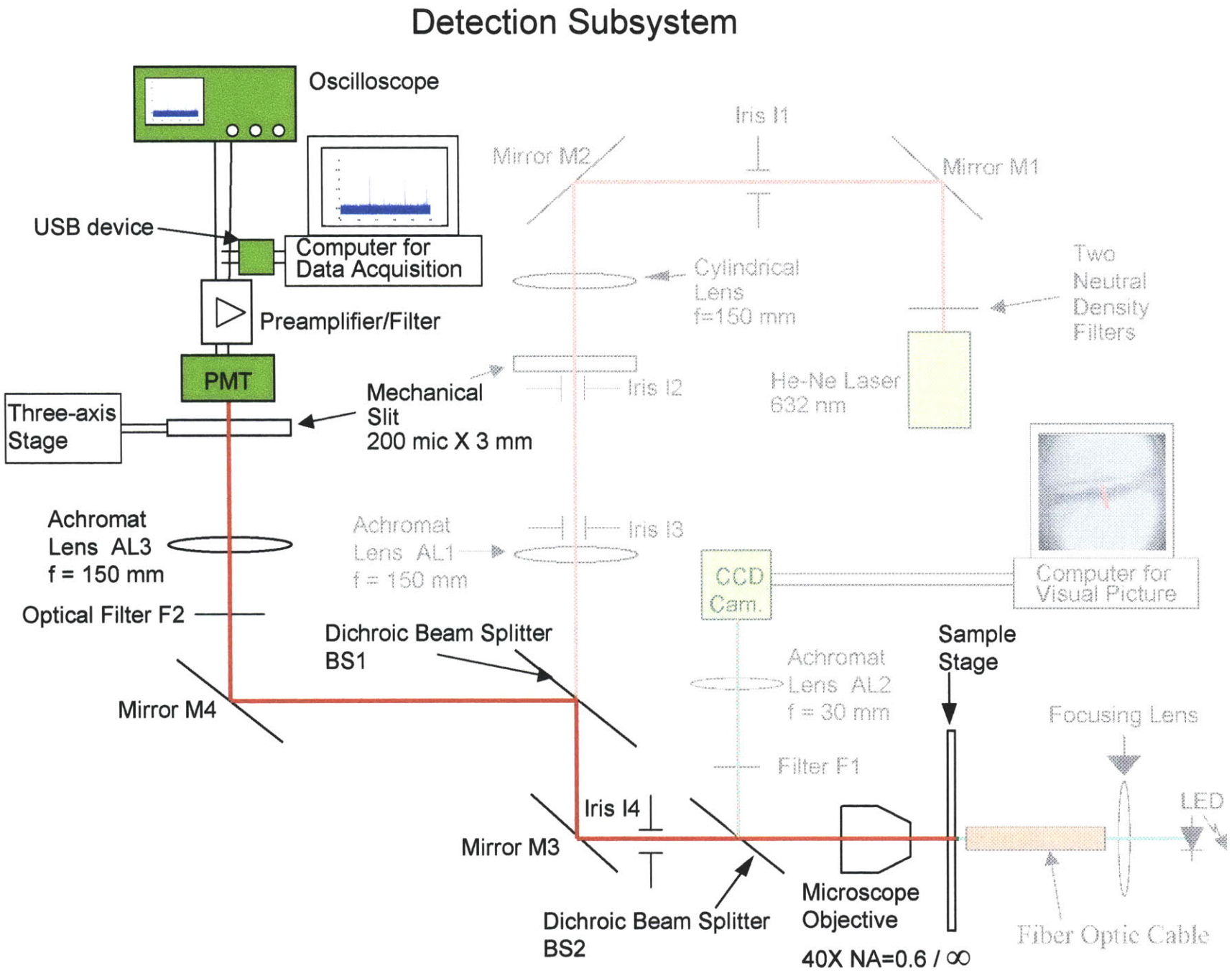

Figure 2.5 Detection Subsystem for the Single-slit, Single-color In Vivo Flow Cytometer

isotopic emission of fluorescence photons, approximately ten percent are captured by the microscope objective.

$NA_{microscope} = 0.6 \Rightarrow \sin(\theta_{\max}) = 0.6 \Rightarrow \theta_{\max} = 36.87^{\circ}$

$\theta_{\max}$ = *half angle of acceptance cone of microscope objective*

$r = 2.50mm$ (*working distance of the microscope objective*)

$y = 2.50mm \cdot \sin(90^{\circ} - 36.87^{\circ}) = 2.00mm$

A = *area of photonemissionsphere intercepted by numerical aperture of microscope objective*

$$A = 2 \cdot \pi \cdot \int_{2.00}^{2.50} \sqrt{r^2 - y^2} \cdot \sqrt{1 + \frac{y^2}{r^2 - y^2}} \cdot dy$$

$$A = 2 \cdot \pi \cdot r \int_{2.00}^{2.50} dy = \pi \cdot r$$

Surface area of sphere $= 4 \cdot \pi \cdot r^2$

$$\textit{Percentage of photons captured} = \frac{\pi \cdot r}{4 \cdot \pi \cdot r^2} \times 100\% = \frac{1}{4 \cdot r} \times 100\% = \frac{1}{10} \times 100\% = 10\%$$

(The actual percentage captured is slightly (although virtually negligible) less than this because emission of photons takes place off the axial centerline of the microscope objective.) These captured photons are channeled by the microscope objective down to dichroic BS2 located below the sampling stage. This dichroic is transmissive to the captured photons. (As mentioned in the previous section, this dichroic transmits light above 600 nm and reflects light below this wavelength). Consequently these photons continue to the mirror below the sampling stage, mirror M3, and are redirected by this mirror to a second dichroic beam splitter, BS2. This second dichroic is not transmissive to the captured photons (this dichroic transmits below and reflects above 650 nm) and, consequently, reflects the fluorescence photons. The captured photons are redirected by this second dichroic to another mirror, mirror M4, which directs them into the leg of the experimental setup where the detector is located. Here, to remove a large percentage of the extraneous light (such as reflected laser light, stray light from the environment, and autofluorescent light), the photons travel through an optical filter (F2) that heavily attenuates light below 650 nm and above 700 nm. This filtered light is then focused by a 150-mm-focal-length achromat lens (AL3) onto a 200 micron by 3 mm mechanical slit, which is confocal with the slit of laser light spanning the blood vessel. A Hamamatsu R3896 photomultiplier tube with extended red sensitivity situated directly behind the mechanical slit is used to detect the fluorescence signal. The photomultiplier tube

converts the photons into a current and then amplifies the current. A Hamamatsu C6271 high voltage power supply socket converts this amplified current to a voltage, and then electronically lowpass filters the voltage signal (bandpass region of the electrical filter is from DC to 10 kHz).

The spectral response of the R3896 (i.e., the current produced by the photomultiplier tube as a function of wavelength) indicates that for the bandpass region of the optical filter in front of the detector (650 nm to 700 nm), the photomultiplier tube produces approximately 62 mA of current for each watt of light power striking the cathode of the device. The amplification of the resulting current, contingent upon the acceleration voltage between the electron multiplying stages of the photomultiplier tube, is a gain of $1X10^6$. (The electron multiplying stages add electrons to the original pool of electrons created by the fluorescence photons, enhancing the original current created by the photons. An acceleration voltage of 750 volts between the electron multiplying stages is used during data acquisition.) This amplified current is then converted to a voltage with a conversion ratio of 0.3 volts per microamp of current. Thus, the resulting output voltage from the PMT/socket device is given by the equation $V = (0.3V/\mu A)\cdot(1000\mu A/mA)\cdot(1X10^6)\cdot(62mA/W)\cdot(P_{Light})$. Maximum output voltage of the PMT/socket device is 10 volts.

The analog voltage signal from the PMT/socket device is fed into a USB device (Data Translation DT 9804). This USB device is an analog-to-digital converter which can sample the analog voltage signal, brought in by a BNC coaxial cable from the PMT/socket device, at a frequency up to 100 kHz. The digitizer in the device (which converts the analog signal to digital) has 32-bit resolution for the voltage range –10 volts to +10 volt. Consequently, the incoming voltage signal is digitized with a 0.3 millivolt resolution. Software on the computer allows this digitized voltage signal to be fed into a computer for storage and analysis.

The data acquisition software on the computer that has been or is presently being used to acquire and store the voltage signal from the USB device was obtained from the web free of charge (URL address is http://www.datatranslation.com/support/results3_all.asp). The first data acquisition software used on the computer is called Scope. This software allows the user to set the

sampling rate (up to the maximum value capable of the USB device), as well as specify the location of storage of the data on the computer. However, the software does not allow for real-time processing of the data, only post-processing, after completion of the experiment. Consequently, this software was ultimately replaced by software that did allow for real-time processing. This software is called DT Measure Foundry, and is a more advanced version of Scope. In addition to being able to specify sampling frequencies and file storage location, this software allows the user to write their own code within the data acquisition program so that the data can be processed during the experiment.

Analysis of the digitized voltage signal involves smoothing of the voltage trace to remove the high frequency components of the trace, and then analyzing the smoothed data for signal (i.e., for the presence of fluorescently labeled cells). When Scope was the data acquisition software being used, the digitized raw data were stored during the experiment, and then smoothed at a later time after completion of the experiment. The disadvantage of this approach (of acquiring raw data at the sampling frequency and then performing post-processing smoothing), was that large data files needed to be stored. The raw data file and the smoothed raw data file, despite being binary files, were tens of megabytes in size. In addition, the smoothed data file, containing such a large number of points (6 million for one minute of data acquisition at 100 kHz) required tens of minutes of computer time to analyze for signal. To remedy these computer file storage problems, as well as expedite the analysis of smoothed data for signal, it was decided to smooth the sampled data during the data acquisition process (i.e., during the experiment), and to save only intermittent points of the smoothed data, equally spaced in time.

Since the maximum cell velocity expected was 5 mm/sec, and the slit width was 5 microns, the Nyquist sampling theorem indicated that approximately 2000 data points per second (time spacing of $5X10^{-4}$ seconds) needed to be saved. However, before this step was implemented, it was decided to verify that similar labeled cell counting results would be obtained. To this end, numerical tests were performed on data from nine different experiments.

Six data traces acquired at a sampling frequency of 50 kHz, three having a very high cell count and three having a low cell count, and three data traces acquired at a

sampling rate of 100 kHz with a low cell count, were chosen as test cases because of their difference in cell count as well as sampling frequency. The results of the offline smoothing/counting process with no point decimation for each of these traces were as follows:

Run Number	Sampling Frequency (kHz)	Number of Points Used in Averaging Process	Time Length of Data Acquisition (sec)	Cell Count	Length of Time Required to Complete Cell Count (sec)
1	50	50	60	2880	410
2	50	50	60	2648	360
3	50	50	60	2511	480
4	50	50	60	62	42
5	50	50	60	60	30
6	50	50	60	61	35
1	100	100	60	47	105
2	100	100	60	59	80
3	100	100	60	55	110

(As mentioned in the introduction chapter, the code that performs the post-processing smoothing is called binarysmoothingfile.m, and the codes that perform the cell counting are binaryreadingcellcounting_new_John.m and analysis8_newmod54_linear_MF_Scope.m. They are discussed in detail in the next chapter.) Next, the computer code below, to be inserted into DT Measure Foundry for real time smoothing and point decimation, was written. The code allows for specification of the number of points to use in the smoothing process (nsm), as well as the amount of point decimation desired (jump). In the code below, the number of points to use in the smoothing process is chosen to be 100, and the amount of point decimation is chosen to be 24 (i.e., save 1 smoothed point in 25). The parameter p5025 is the row vector containing the raw data points to be smoothed, pp5025 is the column vector of the raw data points obtained by transposing the row vector p5025, and g5025 is the column vector containing every 25^{th} smoothed point

jump = 25
nsm = 100

```
numberofloops = (length(p5025)/jump) – (nsm/jump) + 1
pp5025 = p5025'
for j = 1:1:numberofloops
        lower = (((j-1)*(jump)) + 1)
        upper = (((j-1)*(jump)) + 1) + (nsm-1)
        g5025(j) = mean(pp5025(lower:upper))
end
```

DT Measure Foundry allows specification of the size of the vector p5025. This enables the user to control how many raw data points are smoothed and decimated at one time. This is important since computer speed is affected by this number. Too few points results in too much time used in transferring data, too many points results in too much time required for the code to execute. Computer performance appears optimal for a vector size of 2000.

Thus the row vector p5025 containing 2000 points was continually fed into the code, and the column vector g5025 was continually fed into a file, until all the data points of the data trace being smoothed and reduced in size had been processed by the above code. The smoothed/decimated file was then analyzed for cell signal. The results of this process applied to the above nine data traces is shown below. Included are the cell count and time required for cell counting for the original size file and the reduced size file, and the percentage difference in cell count and cell counting time between the original size file and the reduced size file for each trace

.

Run Number	Sampling Frequency (kHz)	Original Size File Cell Count	Reduced Size File Cell Count	Percentage Difference in Cell Count	Original Time Required to Complete Cell Count (sec)	Reduced Time Required to Complete Cell Count (sec)	Percentage of Original Cell Count Time
1	50	2880	2809	2.46	410	13	3.17
2	50	2648	2595	2.00	360	13	3.61
3	50	2511	2428	3.31	480	12	2.5
4	50	62	60	3.23	42	2	4.76
5	50	60	60	0	30	2	6.67

6	50	61	60	1.64	35	2	5.71
1	100	47	47	0	105	2	1.90
2	100	59	58	1.69	80	2	2.5
3	100	55	55	0	110	2	1.82

As one can see by inspection of the results, the agreement in cell count between the original size file and the reduced size file was excellent. In addition, the cell counting process for the reduced size file required only several seconds to complete for all the data traces. Thus it was decided to do real time smoothing and file size reduction (i.e., point decimation).

Therefore, data from the PMT/socket assembly fed into the USB analog-to-digital device would be transferred from the USB device to the smoothing/decimation computer code in blocks of 2000 points at a time via the software DT Measure Foundry. The data would be averaged and every 20^{th} point would be stored into the column matrix g5025. (Every 20^{th} point instead of every 25^{th} point would be saved to further enhance agreement in cell count between the original size smoothed file and the reduced size smoothed file. The time penalty would be minimal.) The smoothed data in the g5025 column matrix would then be transferred to a dcf file (i.e., a specially formatted binary file that DT Measure Foundry or Scope can open and read), as well as displayed by DT Measure Foundry on the computer monitor. This reduced size file would be analyzed at a later time for cell signal using the codes binaryreadingcellcounting_new_John.m and analysis8_newmod54_linear_MF_Scope.m.

Alignment Procedure

Proper alignment of the components of the experimental setup is imperative for good performance by the single-slit, single-color in vivo flow cytometer, since the fluorescence signal from the labeled cells is minimal in power in comparison to the noise sources. Environmental light, laser excitation light, autofluorescent light are noise sources that are all orders of magnitude larger than the fluorescent signal attempting to be detected. Therefore, precise alignment of the parts comprising the cytometer is necessary to achieve maximum excitation of the fluorescence molecules labeling the cells, as well

as maximum capture of the fluorescence photons from the labeled cells, and maximum attenuation of the light from noise sources. To this end, the following procedure was followed in aligning the single-slit, single-color in vivo flow cytometer:

1) The 30-mm-focal-length achromat lens (AL2) was secured into threaded tubing to be attached to the CCD camera. The achromat was attempted to be positioned in the tubing such that, when the tubing was attached to the camera, the achromat would be located its focal length (i.e., 30 mm) in front of the camera. The tubing was attached to the CCD camera. The output of the CCD camera was fed into a television screen. The CCD camera was pointed at a distant object and the image viewed on the television screen to which the CCD was attached. Since the object was a long distance from the camera, only parallel or quasi-parallel rays from the object would enter the camera. Consequently, if the achromat was its focal length in front of the camera, the image on the television screen of the distant object should be in focus. The image was viewed and, if blurry, the achromat was repositioned. This repositioning of the achromat lens continued until the distant object being viewed was in focus. Once the achromat AL2 was properly positioned, the filter F1 was positioned in front of the achromat and secured into the threaded tubing.

2) The threaded tubing containing the achromat lens (AL2) and optical filter (F1) and interfaced with the CCD camera was then interfaced to the side of the structure underneath the sample stage holding the microscope objective and the dichroic beam splitter BS2. This resulted in the camera and achromat being precisely positioned along the horizontal axis of the support structure underneath the sample stage. The microscope objective did not required alignment because it was threaded into the support structure, which held it securely along the vertical axis (of the support structure).

3) A microscope slide with a dirty top surface was placed on the sample stage, and the green LED was activated. The image of the top surface of the microscope slide provided by the microscope objective, achromat lens AL2, and CCD camera was viewed on the television screen to which the camera was interfaced. If the image of the top surface of the microscope slide was not in focus, it was bought into focus by vertical movement of the sample stage upon which the microscope slide was situated. Thus, since only parallel light from the microscope slide was focused onto the detector of the CCD camera, the top surface of the microscope slide was positioned at the focal plane of the microscope objective. (The focal plane of the microscope objective is the desired location for fluorescent excitation since it results in the gathered fluorescence photons becoming collimated after being collected by the microscope objective.)

4) The dichroic beam splitter underneath the sample stage, BS2, was positioned such that the image of the top surface of the microscope slide was centered on the

television screen.

5) Iris I1 was positioned between mirrors M1 and M2, and irises I2 and I3 were positioned between mirrors M2 and M3. The aperture of the irises was positioned three inches above the optical table. A beam height of three inches was chosen since this was the center height of the mounted components that would be placed in the laser light path.

6) The Helium-Neon laser was activated. Using the support upon which the laser is situated, and mirrors M1 and M2 (see Figure 2.1), the beam of the laser was directed through the aperture of the irises. Iris I1 was closed down to allow only the collimated core of the laser beam to pass. Thus, a laser beam parallel to the surface of the optical table and having a height of three inches above the optical table was obtained.

7) Iris I4 was attached to the support structure (holding the camera, achromat AL2, the microscope objective, and the dichroic beam splitter BS2) at its bottom port. Iris I4 was closed down to a small aperture yielding an opening centered in the middle of the port and, hence, centered about the vertical axis of the support structure.

8) Mirror M3, the mirror underneath the sample stage and support structure, was put into place. The dichroic beam splitter BS1 was positioned at a 45^0 degree angle in front of mirror M3. This mirror was then adjusted such that the He-Ne laser light struck the mirror at its center, and was redirected through the aperture formed by iris I4. The television screen displaying the output of the CCD camera was viewed. If the laser light was perpendicularly striking the microscope slide, the image of the reflection of the laser light off the top and bottom surface of the microscope slide would coincide with one another. (The dichroic beam splitter BS2 is slightly reflective at 632 nm, allowing one to view both of these reflections. The reflection off the top surface of the microscope slide is a sharp circle of light while that off the bottom surface is a larger, diffuse, out-of-focus circle of light.) If the two reflections did not coincide, then mirror M3 had to be readjusted for orientation and position. This adjustment of mirror M3 had to continue until the laser beam was striking the center of mirror M3, traveling through the aperture of iris I4, and perpendicularly striking the microscope slide.

9) A 150-mm-focal-length achromat (achromat lens AL1) was secured into threaded tubing. Also, a 150-mm-focal-length cylindrical lens was secured into threaded tubing. With the cylindrical lens positioned in front of the achromat (i.e., the laser light entered the cylindrical lens first), both pieces of tubing were placed through tubing rings attached to the optical table which elevated the center of the lenses three inches off the table (hence, the laser light was traveling down the axial axis of the lenses). The two pieces of tubing were slid back and forth relative to one another until the image formed after the achromat was a slit of light in focus at 150 mm beyond the achromat, and had parallel light in one direction. (As

expected, the distance between the two lenses was 300 mm, the sum of the focal length of each lens.) The two pieces of tubing were then secured together.

10) The two pieces of tubing were then slid towards the dichroic BS1. Viewing the television screen displaying the CCD camera output, a position was found which yielded a focused slit rotated 90^0 from the one formed in step (8), with parallel light in one direction. (That light was parallel in one of the directions of the slit could be verified by moving the sampling stage vertically and seeing that one of the dimensions of the slit stayed constant.) This meant that the 150-mm-focal-length achromat AL1 was now positioned 150 mm from the back focal plane of the microscope objective.

11) The tubing containing the cylindrical lens and achromat AL1 were secured in the tubing rings through which they were slid. They were then detached from one another. A mechanical slit 200 microns by 3 mm was placed 150 mm beyond the cylindrical lens (at the point of sharpest focus of the slit of light formed by the cylindrical lens). This mechanical slit sharply defined the dimensions of the slit of light formed by the cylindrical lens. (The slit of light formed by the cylindrical lens was the object that was imaged by the achromat AL1 at the back focal plane of the microscope objective in step (10).) Iris I2 was repositioned immediately after the mechanical slit to provide a means of controlling the length of the excitation slit at the test sample stage.

12) The mechanical slit in the detector leg was secured to a three-axis stage.

13) The achromat AL3 in the detector leg was secured into threaded tubing. The threaded tubing was slid into tubing rings situated in front of the mechanical slit attached to the three-axis stage. The tubing rings allowed for the sliding along the axial axis of the tubing.

14) An iris was secured to the front and back of the threaded tubing. Both irises were closed down to small apertures.

15) The slit of light formed at the sample stage was partially reflected from the top surface of the microscope slide and imaged to the detector leg. The mirror in the detector leg, M4, was positioned such that the slit of light passed through the aperture of both irises. (This would yield maximum collection by the achromat AL3 of the fluorescence photons originating at the sample stage.)

16) The irises were removed from the tubing. The tubing with the achromat was slid back and forth until the sharpest image of the light slit from the sample stage formed at the plane of the mechanical slit. The tubing was secured in place.

17) The mechanical slit (in the detector leg) was repositioned such that the light slit passed through the mechanical slit. Thus, the mechanical slit in the detector leg was now confocal with the slit of light formed at the sample stage.

18) The bandpass filter F2, used to reduce or eliminate light noise sources, was secured into the tubing containing achromat AL3. The filter was placed in front of the achromat (i.e., the light entering the threaded tubing to be collected by the achromat and focused onto the mechanical slit first passed through the bandpass filter) to avoid compromising the focusing of the light by the achromat.

19) The PMT was placed immediately behind the mechanical slit to maximize the collection by the detector of the photons passing through the mechanical slit, as well as to prevent extraneous light from entering the detector from the side.

Once the system was aligned, data acquisition using the device was begun. The experiments that were performed and the results obtained are discussed in the last chapter. Figure 2.6, shows the axial response curve of the single-slit, single-

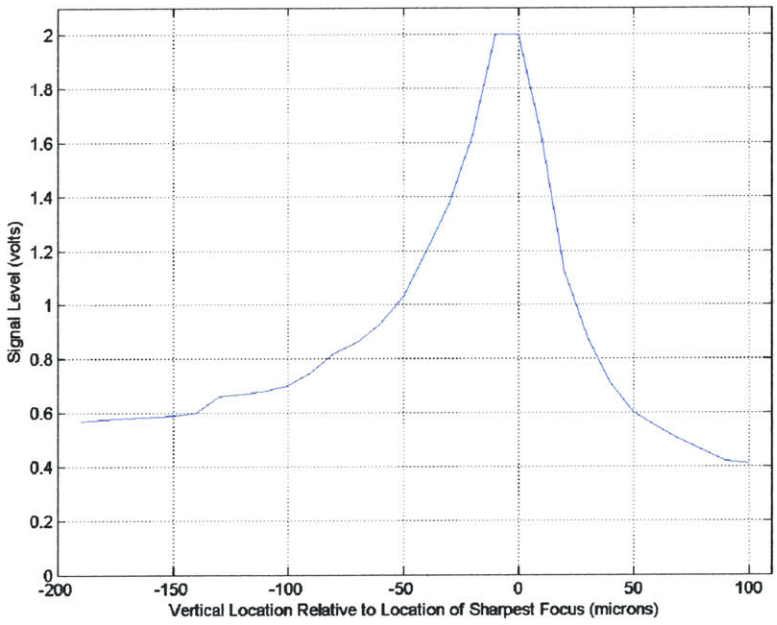

Figure 2.6 Axial response curve of single-slit, single-color system

color system after alignment, obtained by reflecting the excitation laser light off the top surface of a microscope slide situated on the sample stage, and replacing the optical filter F2 in front of the PMT with a slightly less attenuating filter to allow the reflected laser light to reach the detector. The microscope slide was moved from -190 microns below the plane of focus to +100 microns above the plane of focus. The curve obtained shows

that the laser light intensity of the excitation slit is fairly uniform for vertical lengths up to 40 microns, with the light intensity at the upper and lower edges at 75% of the peak excitation intensity at the plane of focus. The asymmetry of the curve is probably due to aberrations from the optics of the fluorescent excitation section.

Two-slit, Two-color System

The two-slit, two-color system is a more advanced version of the single-slit, single-color system. This device can provide one or two laser slits for fluorescence excitation of labeled cells, as well as one or two channels of detection. The excitation slits can be either single color or multicolor. The two excitation wavelengths from which to choose are 473 nm and 632 nm. As mentioned in the introduction, 473 nm was chosen because of its capability to excite cells expressing the EGFP gene, and 632 nm was chosen because of its tissue penetrating and excitation capabilities. The two-slit, two-color system is shown in Figure 2.7. Like the single-slit, single-color system, it is comprised of a transillumination section, a fluorescence excitation section, and a detection section.

Transillumination Section

The transillumination section is shown in Figure 2.8. As in the single-slit, single-color system, an artery or vein of appropriate diameter in the ear of the animal is identified for data acquisition by transilluminating the ear vasculature with a green light emitting diode (LED), and imaging it using a 40X, 0.6 numerical aperture, infinity corrected objective and an achromat lens onto a CCD camera, whose output is fed into a computer monitor for display. The power output of the LED is variable, and is controlled by the circuit shown in Figure 2.3. The dichroic beam splitter, BS3, is used to reflect the transmitted light towards the CCD. This dichroic beam splitter is a band-pass filter that turns on at 410 nm and turns off at 750 nm, and has a transmissivity of 95% between these two wavelengths.

Initially, this beam splitter was a long-pass filter that turned on at 465 nm, and the LED was a blue-emitting diode (peak emission at 460 nm). This combination of LED and dichroic were initially employed because, for the experiments planned, the

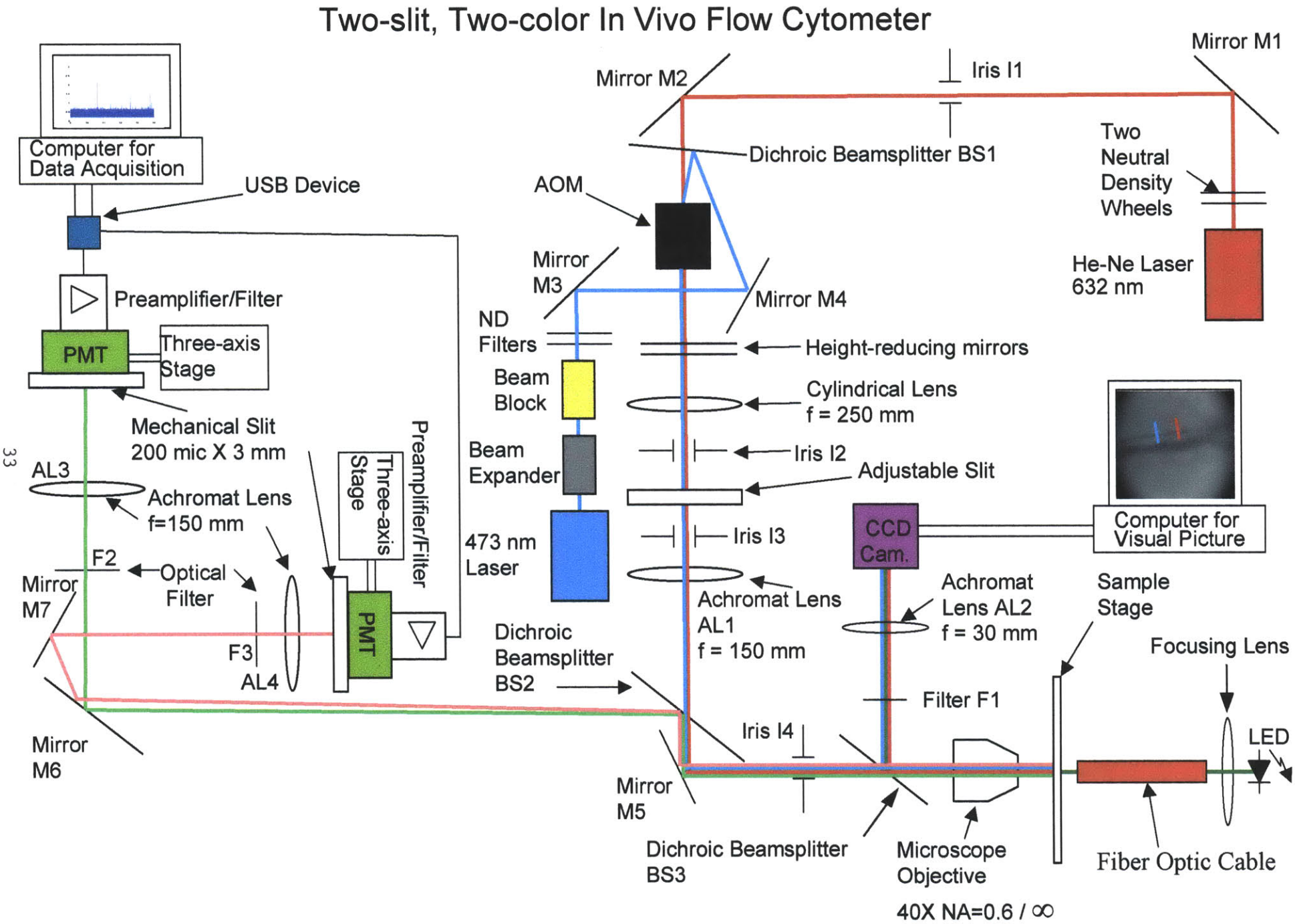

Figure 2.7 The Two-slit, Two-color In Vivo Flow Cytometer

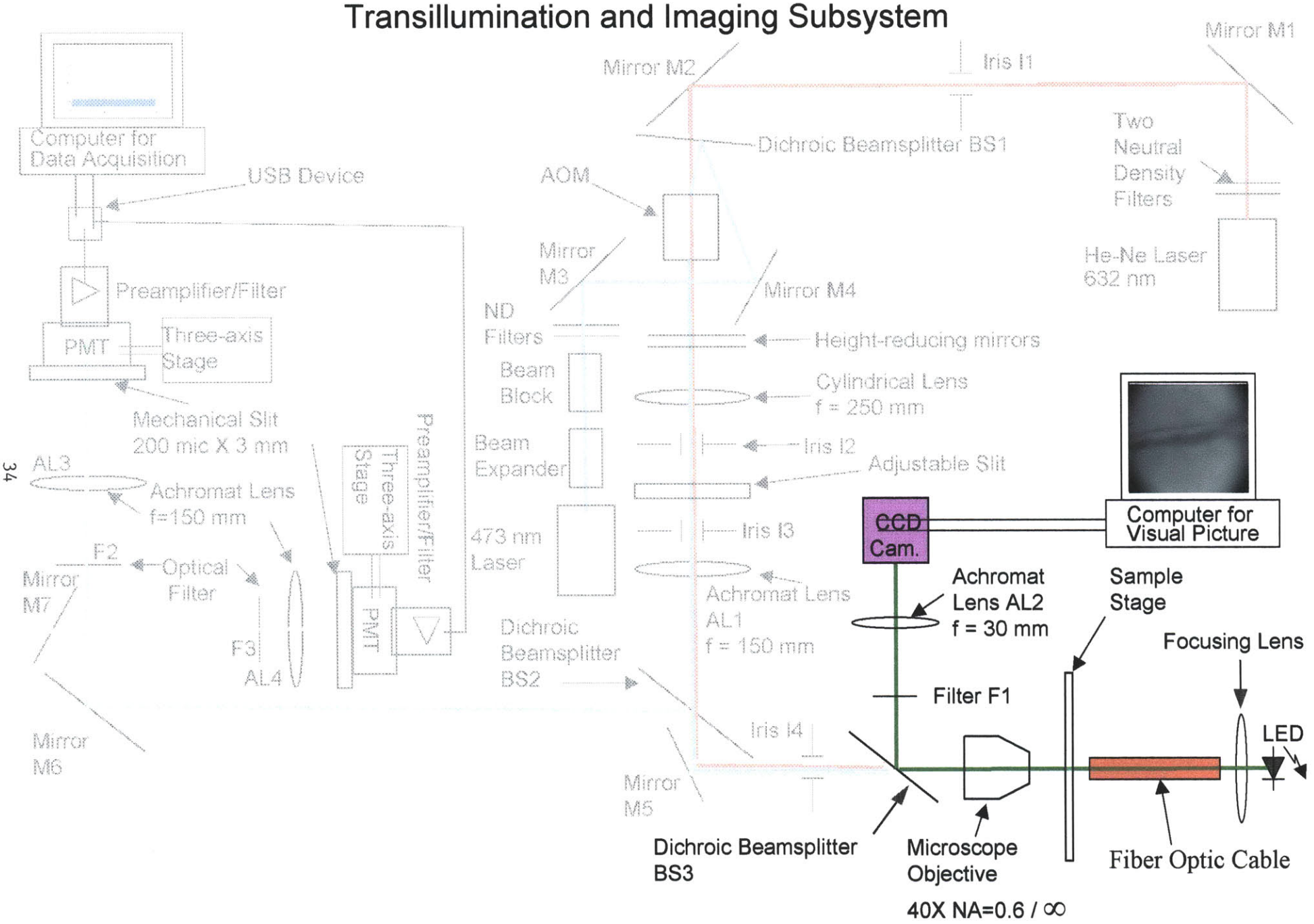

Figure 2.8 Transillumination Subsystem of Two-slit, Two-color Cytometer

fluorescence that was going to occur due to 473 nm excitation was in the green wavelength region, requiring the dichroic to be very transmissive in this area of the electromagnetic spectrum (so that a sufficient number of fluorescence photons would reach the blue channel detector.) However, the blue light did not provide good contrast between vasculature and nonvasculature tissue, making it difficult to find a blood vessel of proper size. Consequently, the dichroic was changed to the band-pass filter now being used in the setup, and the blue LED was replaced with a near infrared LED. Similar to the blue LED, the infrared LED also did not provide enough contrast between blood vessels and surrounding tissue. As a last resort, a green LED was put into the system to see whether sufficient light would reach the CCD camera to form a usable image. Due to the capability to control the power output of the LED, as well as sufficient sensitivity of the CCD camera at this wavelength, this proved to be the case, resulting in an image from which easily a good data acquisition site could be chosen. (The actual sensitivity of the CCD camera was not known a priori because the sensitivity plot supplied with the camera was a normalized curve.)

Like the single-slit, single-color system, the microscope objective and the achromat lens together provide 6.67X magnification, with a field-of-view of 800 μm X 1000 μm and a resolution of 0.43 μm. The pixel size of the CRT upon which the image from the CCD camera is displayed is 1.47 microns. Also, similar to the previous system, the light from the LED is brought to the test sample stage via a fiber optic cable.

Fluorescence Excitation Section

As mentioned above, the fluorescence excitation section can provide two excitation wavelengths, 473 nm and 632 nm. In addition, one or two excitation slits can be chosen. For the two-excitation slit option, both slits can be the same wavelength or each slit can be a different wavelength.

The fluorescence excitation section, shown in Figure 2.9, is comprised of a Helium-neon laser, a blue diode laser, five mirrors (mirrors M1 through M5), an acoustic-optical modulator (AOM), a sample stage, three dichroic beam splitters (BS1 through BS3), a microscope objective, a cylindrical lens, four irises (I1 through I4), an adjustable slit, a beam expander, two neutral density filter structures (each comprised of two wheels

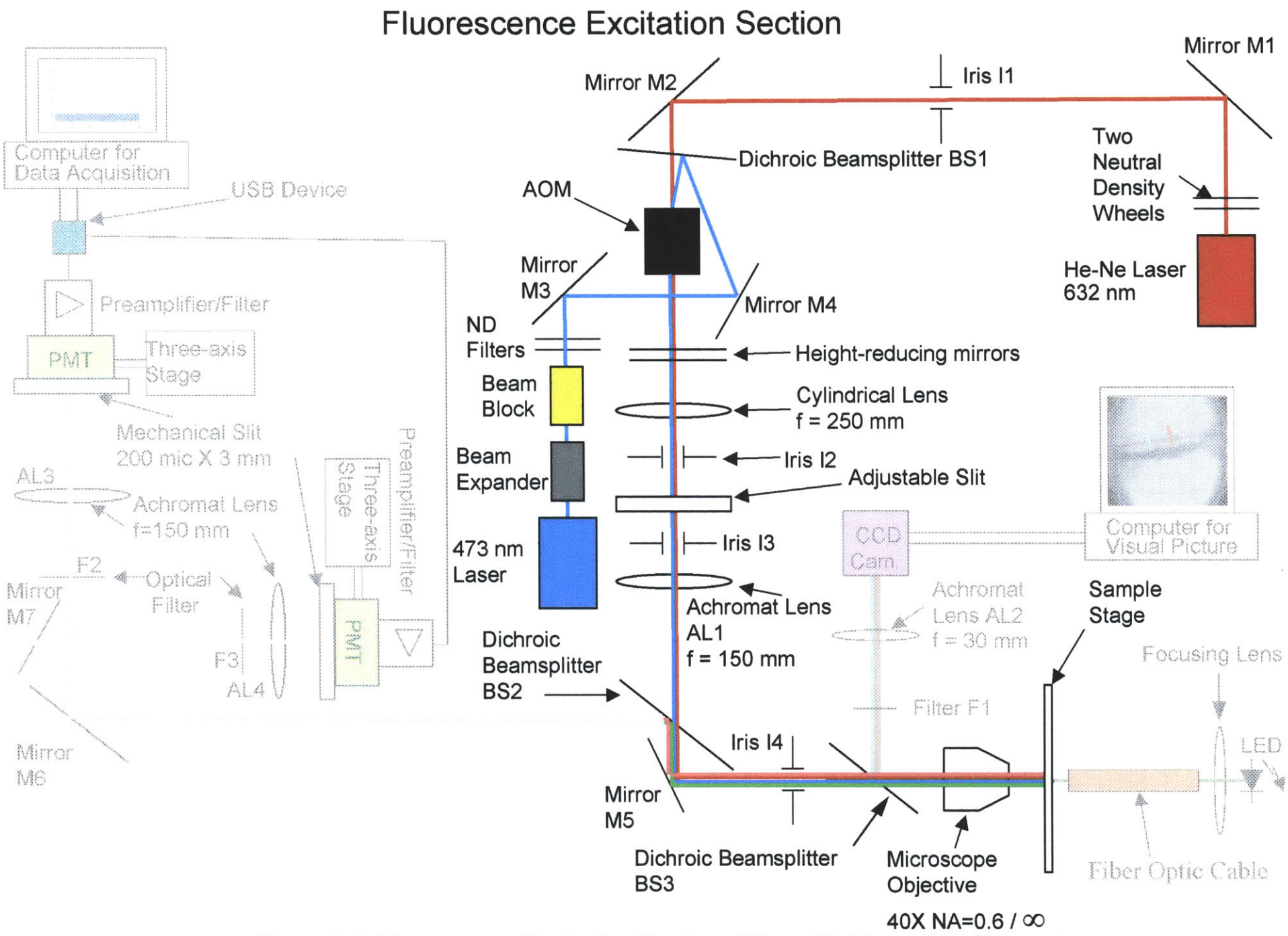

Figure 2.9 Fluorescence Excitation Section of Two-slit, Two-color Cytometer

holding five attenuating filters each), and a beam blocker. The Helium-neon laser provides the 632 nm excitation wavelength. The power output of this laser is controlled by two neutral density filter wheels situated in front of the laser. After passing through the two filter wheels, mirror M1 directs the laser light through iris I1, which spatially filters the light so that only the collimated core of the beam is transmitted, and onto mirror M2. (Like the Helium-neon laser of the single-slit, single-color system, a weak spatially incoherent component is emitted requiring spatial filtering.) Mirror M2 directs the 632-nm laser light through the dichroic beam splitter BS1 and into the AOM. (The dichroic beam splitter BS1 is a long-pass filter that is transmissive at 600 nm and above and is used to combine the 473 nm laser beam with the 632 nm laser beam at the entrance aperture of the AOM.)

The AOM is situated on a rotatable stage to help in positioning the AOM. It is desired that the laser beam enter the AOM at the Bragg angle. The Bragg angle is given by the equation $\theta_B = \frac{\lambda_o \cdot F}{n \cdot v}$, where θ_B is the Bragg angle, λ_o is the wavelength of the laser beam, F is the RF frequency of the EM signal used to generate the acoustic wave which causes the diffraction of the laser beam, n is the index of refraction of the crystal in which the acoustic wave is created, and v is the acoustic wave velocity. At this angle of entrance by the laser light, with the AOM on, maximum diffraction efficiency into the first diffraction order results. Consequently, most of the power from the laser beam will be distributed in the 0^{th} order and 1^{st} order laser light output beams. (All of the power from the laser beam would be in these two orders if the AOM was operating entirely in Bragg regime. An AOM is operating in the Bragg regime if the quality factor Q, equal to $\frac{2 \cdot \pi \cdot \lambda_o \cdot L}{n \cdot \Lambda^2}$, where L is the distance the laser beam travels through the acoustic wave, Λ is the acoustic wavelength, and the other parameters are specified above in the Bragg angle equation, is greater than 4π. This AOM had a Q of 2.7π, meaning that it operated in a transition region which resulted in diffraction orders greater than one being present.) The intensity of the RF signal which creates the acoustic wave in the crystal of the AOM is used to control the power intensity distribution between the two orders, and is adjusted so that the power in the 0^{th} order and 1^{st} order are equal, resulting in equivalent excitation power of the two slits at the sample stage. The angle between the two orders is equal to

twice the Bragg angle. The AOM is on if one desires two-slit, single-color excitation.

With the AOM off, the laser light passes through the device without being diffracted. The AOM is off if one desires single-slit, single-color excitation, or two-slit, two-color excitation. This ability to operate in either a diffracting or nondiffracting mode, as well as the ability to control power distribution in the orders and angle of divergence of the orders, is the reason why an AOM (and not a grating) is used to diffract the laser beam.

After passing through the AOM, the laser light is reduced to the desired elevation of three inches above the optical table by two mirrors. The laser light is initially higher than three inches because the entrance height of the AOM is elevated due to the AOM being situated on a rotatable stage. The height-reducing mirrors direct the laser light into a 250-mm-focal-length cylindrical lens (Melles Griot 01 LCP 135). The cylindrical lens converts the laser light into a vertical slit of light (i.e., the slit of light is perpendicular to the surface of the optical table) by focusing the light only in the horizontal direction. This lens is situated 250 mm from the diffraction origin of the AOM. The cylindrical lens is situated at this location (i.e., one focal length of the cylindrical lens in front of the diffraction origin of the AOM) so that, when the AOM is activated, the laser beams from the AOM will be parallel after traveling through the cylindrical lens. The diffraction origin of the AOM was determined by activating the AOM and following the beams of light back to their point of intersection in the AOM. After traveling through the cylindrical lens, the light is directed through two irises (I2 and I3) and an adjustable slit. The two irises provide alignment guidance and spatial filtering, helping to ensure that the laser light is following the right trajectory and blocking out extraneous light (from the environment and higher diffraction orders when the AOM is activated) traveling along the laser light path. The adjustable slit, located 250 mm beyond the cylindrical lens (at the point of sharpest focus of the cylindrical lens) provides control over slit length at the sample stage. (The adjustable slit, which has a rectangular aperture, is used instead of an iris, which has a circular aperture, because the slits of excitation light, being spatially separated, are not centered along the path of light travel but are symmetric about the path of light travel.) As in the single-slit, single-color system, keeping the slit length to the diameter of the blood vessel at the data acquisition site is imperative to keep the noise

source contribution of autofluorescence by nonvasculature tissue to a minimum. The 150-mm-focal-length achromat lens AL1, situated 150 mm beyond the adjustable slit, Fourier transforms, via mirror M5, the slit of light created by the cylindrical lens to the back focal plane of the microscope objective. The beam splitter between achromat AL1 and mirror M5, BS2, is a neutral density filter with an OD of 0.6 (25% transmission) at 632 nm (and 473 nm). The dichroic beam splitter between mirror M5 and the microscope objective, BS3, has 95% transmission at 632 nm (and 473 nm). The microscope objective Fourier transforms the slit of light at its back focal plane across the blood vessel at the chosen data acquisition site. (Hence, as in the single-slit, single-color cytometer, the achromat and microscope objective together image at the chosen data acquisition site the slit of light created by the cylindrical lens.) The in vivo flow cytometer is aligned such that this chosen acquisition site is at the front focal plane of the microscope objective.

The 473-nm excitation light is provided by a diode-pumped solid state laser (Melles Griot 85 BCA 015). A beam expander at the exit aperture of the diode laser is used to adjust the divergence of the beam so that the slit of excitation light from this laser focuses at the same plane at the test sample stage as the excitation slit from the Helium-neon laser. Like the Helium-neon laser, the power of the excitation beam is controlled by two neutral density wheels. A beam blocker was constructed and is situated in front of the neutral density wheels to block the reflected light from the neutral density filters (the neutral density filters attenuate by reflection and not absorption). This beam blocker is necessary for this laser because, contrary to the Helium-neon laser which is situated at the back of the setup, this laser is situated at the front of the optical table. Therefore, the beam blocker is necessary to avoid exposing users of the system to reflected laser light. Mirrors M3 and M4 direct the laser light to dichroic BS1, which reflects the light into the entrance aperture of the AOM. Beyond this point, the 473-nm laser light encounters the same optical equipment as the 632-nm laser light.

Thus to obtain a single slit of excitation light at 473 nm, only the diode laser is turned on (the Helium-neon laser and AOM are off.) A single slit at 632 nm is obtained by activating the Helium-neon laser only. A double slit of 473 nm laser light is obtained by activating the diode laser and the AOM. Similarly, a double slit of 632 nm laser light

is obtained by activating the Helium-neon laser and AOM. Lastly, a 473 nm slit and a 632 nm slit (i.e., two-slit, two-color) is obtained by activating both lasers with the AOM off. (Note that other scenarios, such as two slits at 632 nm and one at 473 nm, or two slits at 473 nm and one at 632 nm, are possible with further adjustment of the angular orientation of the AOM relative to the incoming laser light.)

Detection Section

The detection section for the two-slit, two-color system is similar to the single-slit, single-color system, except now two detector legs are present. The detection section for the two-slit, two-color system is shown in Figure 2.10. Like the single-slit, single-color system, the 0.6-numerical-aperture microscope objective gathers about10% of the fluorescence photons and channels them to the mirror (M5) below the sample stage. Because these photons originate at the front focal plane of the microscope objective, the photons of each point fluorescence remain parallel to one another as they travel, allowing focusing of the photons (by a lens) to take place at a remote location from their point of origin without any losses due to divergence. The mirror, M5, redirects them to the beam splitter BS2, which reflects approximately 75% of the fluorescence photons to mirror M6. For the scenario of a double, nonoverlapping excitation slits, where the slits are not overlapping at the point of data acquisition, the fluorescence photons created for an excitation at each slit follow two nonparallel paths of travel to mirror M6.

The angle off the vertical that the photons leave the microscope objective, α (see Figure 2.11), is determined by the radial distance, d, the fluorescence photons originate from the axial axis of the microscope objective. The exit angle of the photons is given by the equation $\alpha = \tan^{-1}(40 \cdot d/180) = \tan^{-1}(d/4.5)$, where 40 is the magnification of the microscope objective when the photons originating at the front focal plane of the microscope objective are focused by an achromat with a focal length of 180 mm, and 4.5 is the back focal length (in mm) of the microscope objective. Thus, the radial distance of displacement of the photons from their location at the exit of the microscope objective is given by the equation $s = l \cdot \tan(\alpha)$, where l is the distance traveled by the photons from the principal exit plane of the microscope objective (assumed to be located at the exit of the microscope objective) to the location of interest measured along the path of a photon

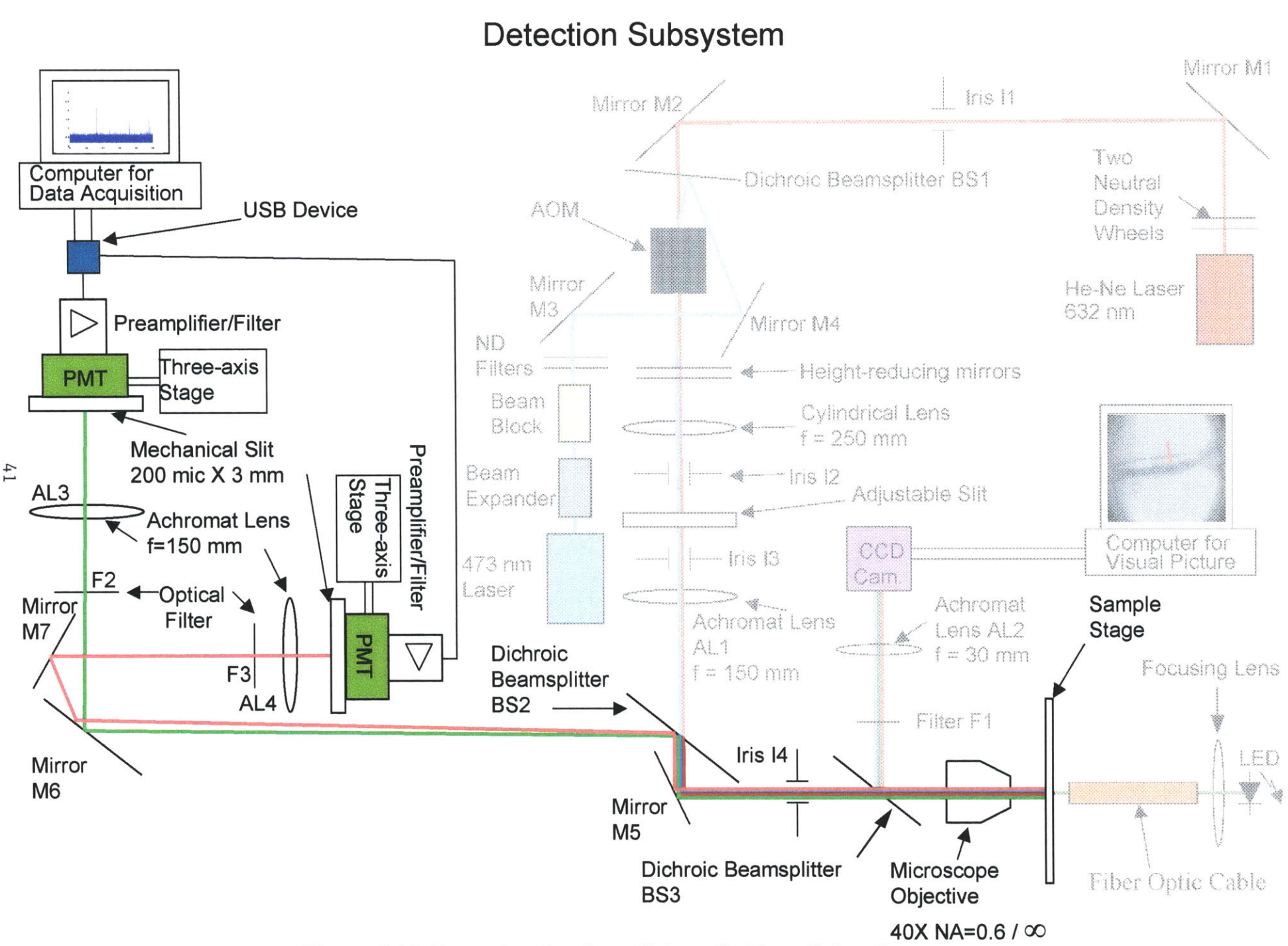

Figure 2.10 Detection Section of Two-slit, Two-Color Cytometer

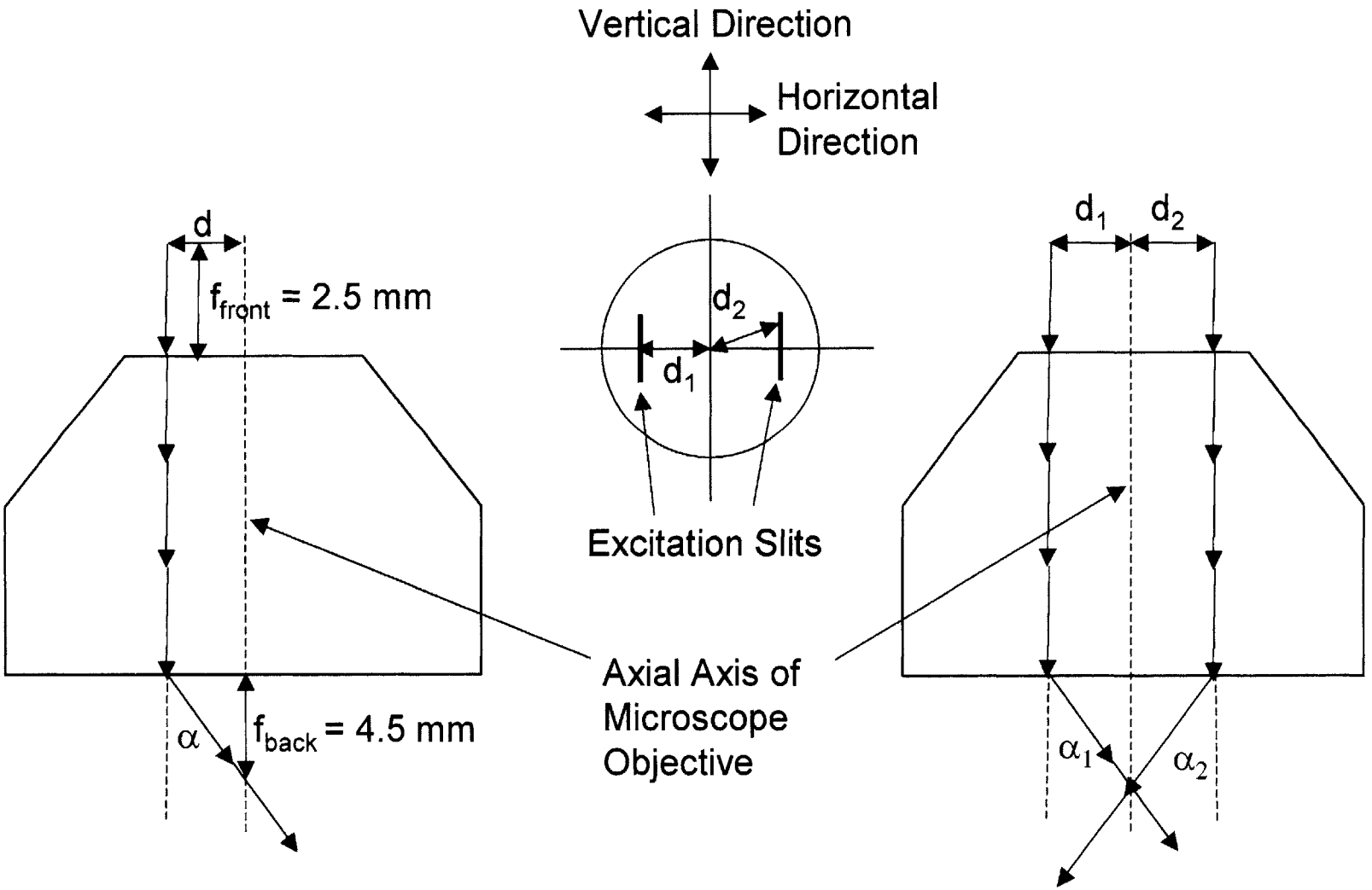

Figure 2.11 Microscope Exit Angle α of Fluorescence Photons

originating at the axial centerline of the microscope objective, and s is the distance of displacement of the photons in the radial direction from their location at the exit of the microscope objective. (Note that for the large values of l of concern below, any error in the location of the principal exit plane of the microscope objective introduces negligible error in the final answer.) The radial displacement can be decomposed into a horizontal and vertical displacement. Choosing the direction perpendicular to the long dimension of the excitation slits to be horizontal, and the direction parallel to the long dimension of the excitation slits to be vertical, and knowing that the angle between the horizontal and vertical components is conserved during the displacement process, one can calculate the horizontal and vertical displacement between the two streams of photons (created by an excitation at each slit). Thus if the excitation slits are positioned 50 microns apart in the horizontal direction (i.e., each excitation slit is 25 microns from the plane containing the axial axis of the microscope objective and oriented parallel to the long dimension of the slits), then the horizontal separation between two streams of photons at distance l (as defined above) millimeters from the exit plane of the microscope objective is given by the equation $h = [2 \cdot l \cdot (25\mu)/(4.5mm)] - 50\mu$, where a negative h means that the centerline photons have not crossed each others path (which will occur for l less than the back focal length, 4.5 mm, of the microscope objective).

The slits at the test sample stage are oriented such that the horizontal separation direction of two streams of photons (originating at separate excitation slits) is parallel to the surface of the optical table upon which the experimental setup is mounted. Thus, at mirror M6, located approximately 400 mm from the exit plane of the microscope objective, the horizontal separation of the centerline of two streams of photons originating at separate excitation slits is 4.39 mm. Therefore, since the aperture diameter of the microscope objective is 5.4 mm (hence, the maximum diameter of each stream of photons), the mirror M6 had to have at least a 9.79 mm horizontal width to reflect all photons from both streams. In addition, if each slit height is 50 μ, the vertical separation of the centerlines of the two streams of photons is given by the same equation above, yielding a centerline to centerline separation of 4.39 mm, and, hence, a required vertical height of 9.79 mm at 4.39 mm to the right and left of the vertical centerline of the mirror. {Note that the actual equation is

Vertical separation of the centerline of the two columns of photons = (l) [(length of slit in mm)/4.5mm)] – (length of slit in mm).}

Consequently, a mirror 25.4 mm in diameter was used (and properly oriented) to guarantee reflection of all fluorescence photons.

The two streams of fluorescence photons reflected off mirror M6 are each directed to a different location. This can be done because each stream of photons strike the mirror M6 at a different angle. One stream of the fluorescence photons is directed to optical filter F2, which filters out extraneous light and transmits the fluorescence photons to achromat lens AL3 for focusing onto a 200 micron by 3 mm mechanical slit (confocal with the excitation slit at the test sample stage that produced the fluorescence photons) and detection by a PMT. The other stream of photons is directed to mirror M7, which reflects the fluorescence photons to another detection leg for filtering, focusing, and detection. At both detection legs, located approximately 900 mm from the exit of the microscope objective, the physical dimensions of the filters and lens required to pass all photons of a 5 micron by 50 micron slit and a 5.4 mm microscope exit aperture are 6.40 mm by 15.35 mm. Consequently, filters F2 and F3, and achromats AL3 and AL4 are 25.4 mm in diameter. In addition, note that at mirror M7, located approximately 780 mm from the exit plane of the microscope objective, the horizontal separation of the centerlines of the two streams of photons is 8.62 mm, allowing for complete separation and detection of all fluorescence photons of each stream of photons.

The analog voltage signals from the photomultiplier tubes are sent to an analog-to-digital converter (DT 9806 made by Data Translation), which samples each analog PMT voltage signal at 25 kHz and digitizes with 64-bit resolution. These digitized signals are stored in a buffer in the DT9806 and then sent to a computer, in blocks of 2000 points, to be processed. The data acquisition software being used, which controls the acquisition of data, the transfer of data within the computer, as well as the flow of data between the analog-to-digital converter and the computer, is DT Measure Foundry. The computer code shown below was written and incorporated into DT Measure Foundry to remove the high frequency components of the data (via averaging) and to save every fifth point of smoothed data from each channel (i.e., from each digitized PMT voltage trace). Twenty-five point smoothing is employed.

```
jump = 5
nsm = 25
numberofloops = (length(p5025)/jump) – (nsm/jump) + 1
pp5025 = p5025'
for j = 1:1:numberofloops
        lower = (((j-1)*(jump)) + 1)
        upper = (((j-1)*(jump)) + 1) + (nsm-1)
        g5025(j) = mean(pp5025(lower:upper))
end
pppp5025 = ppp5025'
for jj = 1:1:numberofloops
        lower = (((jj-1)*(jump)) + 1)
        upper = (((jj-1)*(jump)) + 1) + (nsm-1)
        gg5025(jj) = mean(pppp5025(lower:upper))
end
```

The row vectors p5025 and ppp5025 contain the digitized data from each photomultiplier tube. The smoothed data for each data acquisition channel are stored in the column vectors g5025 and gg5025. Once the 2000 points in each row vector are processed by the above code, the contents of the column vectors g5025 and gg5025 are each written to an independent file, as well as displayed on the computer screen as a function of time in a scrolling format. The process of transferring data in 2000-point blocks for each channel, smoothing and saving every fifth point, and then storing and displaying the data, is repeated until the experiment is finished. The smoothed files created are analyzed for labeled cell presence off-line using the codes binaryreadingcellcounting_new_John.m and analysis8_newmod54_linear_MF_Scope.m.

Alignment Procedure

Like the single-slit, single-color system, proper alignment of the flow cytometer components for the two-slit, two-color system is imperative for excitation and detection of in vivo circulating labeled cells. Indeed, for two channel excitation and detection, alignment is even more critical to ensure, not only minimization of noise contributions and maximization fluorescence signal, but also that interference does not occur between the two channels. Two different alignment procedures were used, with a variation on them depending on whether one was aligning for two slits at wavelength 632 nm, two slits at wavelength 473 nm, or one slit at wavelength 632 nm and the other slit at

wavelength 473 nm. Each alignment procedure had its advantages and disadvantages. For example, Alignment Procedure 1 required overall less movement of the parts but more adjustment of the mirrors, whereas for Alignment Procedure 2, the opposite was true.

The alignment procedures are as follows:

Alignment Procedure 1

I) Two slits at wavelength 632 nm

1) The 30-mm-focal-length achromat lens (AL2) was secured into threaded tubing to be attached to the CCD camera. The achromat was attempted to be positioned in the tubing such that, when the tubing was attached to the camera, the achromat would be located its focal length (i.e., 30 mm) in front of the camera. The tubing was attached to the CCD camera. The output of the CCD camera was fed into a television screen. The CCD camera was pointed at a distant object and the image viewed on the television screen to which the CCD was attached. Since the object was a long distance from the camera, only parallel or quasi-parallel rays from the object would enter the camera. Consequently, if the achromat was its focal length in front of the camera, the image on the television screen of the distant object should be in focus. The image was viewed and, if blurry, the achromat was repositioned. This repositioning of the achromat lens continued until the distant object being viewed was in focus. Once the achromat AL2 was properly positioned, the filter F1 was positioned in front of the achromat and secured into the threaded tubing.

2) The threaded tubing containing the achromat lens (AL2) and optical filter (F1) and interfaced with the CCD camera was then interfaced to the side of the structure underneath the sample stage holding the microscope objective and the dichroic beam splitter BS2. This resulted in the camera and achromat being precisely positioned along the horizontal axis of the support structure underneath the sample stage. The microscope objective did not required alignment because it was threaded into the support structure, which held it securely along the vertical axis (of the support structure).

3) A microscope slide with a dirty top surface was placed on the sample stage, and the green LED was activated. The image of the top surface of the microscope slide provided by the microscope objective, achromat lens AL2, and CCD camera was viewed on the television screen to which the camera was interfaced. If the image of the top surface of the microscope slide was not in focus, it was bought into focus by vertical movement of the sample stage upon which the microscope slide was situated. Thus, since only parallel light from the microscope slide was focused onto the detector of the CCD camera, the top surface of the microscope

slide was positioned at the focal plane of the microscope objective. (The focal plane of the microscope objective is the desired location for fluorescent excitation since it results in the gathered fluorescence photons becoming collimated after being collected by the microscope objective.)

4) The dichroic beam splitter underneath the sample stage, BS3, was positioned such that the image of the top surface of the microscope slide was centered on the television screen.

5) Iris I1 was positioned between mirrors M1 and M2, and irises I2 and I3 were positioned between the height-reducing mirrors and mirror M5. The dichroic beam splitter BS1, which transmits above 600 and reflects below this wavelength, was positioned at the entrance height and several inches in front of the AOM at approximately a 45^0 angle. The aperture of iris I1 was positioned to the height of the entrance aperture of the AOM, and the aperture of irises I2 and I3 were positioned three inches above the optical table. A beam height of three inches after the AOM was chosen since this was the center height of the mounted optical components that would be placed in the laser light path between the height-reducing mirrors and mirror M5.

6) The Helium-neon laser was activated. Using the support upon which the Helium-neon laser is situated, and the first two mirrors, M1 and M2, the beam of the laser light was directed through iris I1 and into the entrance aperture of the AOM. Iris I1 was closed down to allow only the collimated core of the laser beam to pass. Thus the laser beam of light from the Helium-neon laser entered the AOM parallel to the surface of the optical table.

7) The height-reducing mirrors, situated after the exit aperture of the AOM, were adjusted to direct the laser beam exiting the AOM through the aperture of irises I2 and I3. This resulted in the laser beam being parallel to the surface of the optical table and three inches above the surface.

8) An iris (iris I4) was attached to the support structure (holding the camera, achromat AL2, microscope objective, and dichroic BS3) at its bottom port. Closing down the iris resulted in a small aperture centered in the middle of the port and, hence, centered around the vertical axis of the support structure.

9) Mirror M5, the mirror underneath the sample stage and support structure, was put into place. A dichroic beamsplitter (BS2), which is a filter that transmits the excitation wavelengths of 473 nm and 632 nm and reflects the fluorescence photons to the detector section of the setup, was positioned at a 45^0 degree angle in front of the mirror. The mirror M5 was then adjusted such that the laser beam struck the mirror symmetrically around its center, and were redirected through the aperture formed by iris I4. The television screen displaying the output of the CCD camera was viewed. If the laser light was perpendicularly striking the microscope slide, the image of the reflection of the laser light off the top and

bottom surface of the microscope slide would coincide with one another. (The dichroic underneath the microscope objective, BS3, is slightly reflective at 632 nm [and 473 nm], allowing one to view both of these reflections. The reflection off the top surface of the microscope slide is a sharp circle of light while that off the bottom surface is a larger, diffuse, out-of-focus circle of light.) If the two reflections did not coincide, then mirror M5 had to be readjusted for orientation and position. This adjustment of mirror M5 had to continue until the laser beam was striking the center of mirror M5, traveling through the aperture of the iris, and perpendicularly striking the microscope slide.

10) A 150-mm-focal-length achromat was secured into threaded tubing. Also, a 250-mm-focal-length cylindrical lens was secured into a support. With the cylindrical lens positioned in front of the achromat (i.e., the laser light entered the cylindrical lens first), both lenses were slid back and forth relative to one another until the image formed after the achromat was a slit of light in focus 150 mm beyond the achromat, and had parallel light in one dimension. (As expected, the distance between the two lenses was 400 mm, the sum of the focal length of each lens.) The two lenses were then secured relative to one another to maintain the correct distance (400 mm) between them.

11) The two lenses were then slid towards the dichroic BS2. Viewing the television screen displaying the CCD camera output, a position was found which yielded a focused slit rotated 90^0 from the one formed in step (11), with parallel light in one direction. (That light was parallel in one of the directions of the slit could be verified by moving the sampling stage vertically and seeing that one of the dimensions of the slit stayed constant.) This meant that the 150-mm-focal-length achromat was now positioned 150 mm from the entrance focal plane of the microscope objective.

12) The cylindrical lens and achromat lens (AL1) were secured in place, and detached from one another. An adjustable aperture was placed at the point of sharpest focus between the cylindrical lens and the achromat (i.e., at the back focal plane of the cylindrical lens) to provide a means of controlling the length of the excitation slit at the test sample stage.

13) The AOM was activated. The output at the exit aperture of the AOM was checked to see if there were only two dominant orders present (indicating that the laser beam was entering the AOM at the Bragg angle). If more than two dominant laser beams were present, then mirrors M1 and M2 were adjusted for direction of the beam in the horizontal plane (i.e., the plane parallel to the surface of the table) until only two dominant orders of light were present.

14) The two slits of light formed by the cylindrical lens were checked for parallelism in the path of travel. This condition was necessary for alignment of the slits at the test sample stage. If the paths of travel of the slits were diverging, then the AOM

and cylindrical lens were too close to one another. Conversely, if the paths of travel of the slits were converging, then the AOM and cylindrical lens were too far apart from one another. If the paths of travel of the two slits were not parallel, then the AOM had to be repositioned (the repositioning direction based on whether the paths of travel of the slits of light were diverging or converging), and mirrors M1, M2, M5, and the height-reducing mirrors readjusted until all the alignment requirements stipulated in steps (6) to (14) were satisfied.

15) The achromats AL3 and AL4 in the detector section of the experimental setup, used to focus the fluorescence photons, were each secured into separate threaded tubing. The threaded tubings were slid into tubing rings. The tubing rings allowed for sliding of the tubings along the axial axis of the achromats.

16) A structure comprised of a photomultiplier tube (PMT), a mechanical slit 200 μ by 3 mm (situated in front of the PMT), and a three-axis stage was positioned behind each threaded tubing such that the axial center of the mechanical slit approximately coincided with the axial center of the achromat.

17) An iris was secured to the front and back of each threaded tubing. All the irises were closed down to small apertures.

18) The slits of light formed at the sample stage were partially reflected from the top surface of the microscope slide and directed by the microscope objective, mirror M5, and beam splitter BS2, to mirror M6. This mirror was positioned such that one of the slits of light struck the aperture of the front iris of the tubing directly in front of mirror M6 (i.e., the tubing containing achromat AL3), while the other slit of light struck mirror M7. The height of the tubing containing AL3 was adjusted and mirror M6 reoriented so that the slit of light striking the aperture of the front iris passed through the back iris as well. Mirror M7 was repositioned to intercept the other slit and direct it into the front iris of the tubing containing AL4. The height of the tubing containing achromat AL4 was adjusted and mirror M7 reoriented so that the slit of light directed at the tubing passed through the two irises attached to the tubing. (This alignment was done to maximize the collection by the achromats of the fluorescence photons originating at the sample stage.)

19) The irises were removed from the threaded tubings. The tubings were slid back and forth until the sharpest image of the light slit from the sample stage directed through each tubing formed at the plane of each mechanical slit. The tubings were secured in place.

20) The mechanical slits were repositioned such that the light slit focused on them passed through the mechanical slit. Thus, each mechanical slit in the detection section was now confocal with their excitation slit of light formed at the sample stage.

21) A bandpass filter, used to reduce or eliminate light noise sources, was secured

into each threaded tubing. The filter was placed in front of the achromat (i.e., the light entering the threaded tubing to be collected by the achromat and focused onto the mechanical slit first passed through the bandpass filter) to avoid compromising the focusing of the light by the achromat.

II) Two slits at wavelength 473 nm

If it was desired to create a double slit using the 473 nm laser rather than the 632 nm laser, then the above alignment procedure for two slits at 632 nm is executed with steps (5), (6), (13) and (14) replaced with the following:

5) The beam expander and the beam blocker were positioned between the diode laser and mirror M3, and irises I2 and I3 were positioned between the height-reducing mirrors and mirror M5. The dichroic beam splitter BS1, which transmits above 600 and reflects below this wavelength, was positioned at the entrance height and several inches in front of the AOM at approximately a 45^0 angle. The aperture of the beam blocker was positioned to the height of the entrance aperture of the AOM, and the aperture of irises I2 and I3 were positioned three inches above the optical table. A beam height of three inches after the AOM was chosen since this was the center height of the mounted optical components that would be placed in the laser light path between the height-reducing mirrors and mirror M5.

6) The diode laser was activated. Using the support upon which the diode laser is situated, and the mirrors M3 and M4, the beam of the laser light was directed into the entrance aperture of the AOM. Thus the laser beam of light from the diode laser entered the AOM parallel to the surface of the optical table.

13) The AOM was activated. The output at the exit aperture of the AOM was checked to see if there were only two dominant orders present (indicating that the laser beam was entering the AOM at the Bragg angle). If more than two dominant laser beams were present, then mirrors M3 and M4 were adjusted for direction of the beam in the horizontal plane (i.e., the plane parallel to the surface of the table) until only two dominant orders of light were present.

14) The two slits of light formed by the cylindrical lens were checked for parallelism in the path of travel. This condition was necessary for alignment of the slits at the test sample stage. If the paths of travel of the slits were diverging, then the AOM and cylindrical lens were too close to one another. Conversely, if the paths of travel of the slits were converging, then the AOM and cylindrical lens were too far apart from one another. If the paths of travel of the two slits were not parallel, then the AOM had to be repositioned (the repositioning direction based on whether the paths of travel of the slits of light were diverging or converging), and mirrors M3, M4, M5, and the height-reducing mirrors readjusted until all the alignment requirements stipulated in steps (6) to (14) were satisfied.

III) One slit at wavelength 632 nm and the other at 473 nm

If it is desired to create a two excitation slits with one at 473 nm and the other at 632 nm, then the above alignment procedure for two slits at 632 nm is executed with steps (13) and (14) replaced with the following:

13) The beam expander and the beam blocker were positioned between the diode laser and mirror M3. The aperture of the beam blocker was positioned to the height of the entrance aperture of the AOM. The diode laser was activated. Using the support upon which the diode laser is situated, and the mirrors M3 and M4, the beam of the laser light was directed into the entrance aperture of the AOM. Thus the laser beam of light from the diode laser entered the AOM parallel to the surface of the optical table. The position of the blue slit at the sample stage was viewed on the television screen interfaced with the AOM. Mirrors M3 and M4 were adjusted until the desired location of the slit was achieved. (Note that the beam expander in the blue channel may have to be adjusted to get the blue excitation slit and to focus at the same plane as the red excitation slit.)

14) The two slits of light formed by the cylindrical lens were checked for parallelism in the path of travel. This condition was necessary for alignment of the slits at the test sample stage. If the paths of travel of the slits were diverging, then the AOM and cylindrical lens were too close to one another. Conversely, if the paths of travel of the slits were converging, then the AOM and cylindrical lens were too far apart from one another. If the paths of travel of the two slits were not parallel, then the AOM had to be repositioned (the repositioning direction based on whether the paths of travel of the slits of light were diverging or converging), and mirrors M1 through M5, and the height-reducing mirrors readjusted until all the alignment requirements stipulated in steps (6) to (14) were satisfied.

Alignment Procedure 2

I) Two slits at wavelength 632 nm

1) The 30-mm-focal-length achromat lens (AL2) was secured into threaded tubing to be attached to the CCD camera. The achromat was attempted to be positioned in the tubing such that, when the tubing was attached to the camera, the achromat would be located its focal length (i.e., 30 mm) in front of the camera. The tubing was attached to the CCD camera. The output of the CCD camera was fed into a television screen. The CCD camera was pointed at a distant object and the image viewed on the television screen to which the CCD was attached. Since the object was a long distance from the camera, only parallel or quasi-parallel rays from the object would enter the camera. Consequently, if the achromat was its focal length in front of the camera, the image on the television screen of the distant object should be in focus. The image was viewed and, if blurry, the achromat was repositioned. This repositioning of the achromat lens continued until the distant

object being viewed was in focus. Once the achromat AL2 was properly positioned, the filter F1 was positioned in front of the achromat and secured into the threaded tubing.

2) The threaded tubing containing the achromat lens (AL2) and optical filter (F1) and interfaced with the CCD camera was then interfaced to the side of the structure underneath the sample stage holding the microscope objective and the dichroic beam splitter BS2. This resulted in the camera and achromat being precisely positioned along the horizontal axis of the support structure underneath the sample stage. The microscope objective did not required alignment because it was threaded into the support structure, which held it securely along the vertical axis (of the support structure).

3) A microscope slide with a dirty top surface was placed on the sample stage, and the green LED was activated. The image of the top surface of the microscope slide provided by the microscope objective, achromat lens AL2, and CCD camera was viewed on the television screen to which the camera was interfaced. If the image of the top surface of the microscope slide was not in focus, it was bought into focus by vertical movement of the sample stage upon which the microscope slide was situated. Thus, since only parallel light from the microscope slide was focused onto the detector of the CCD camera, the top surface of the microscope slide was positioned at the focal plane of the microscope objective. (The focal plane of the microscope objective is the desired location for fluorescent excitation since it results in the gathered fluorescence photons becoming collimated after being collected by the microscope objective.)

4) The dichroic beam splitter underneath the sample stage, BS3, was positioned such that the image of the top surface of the microscope slide was centered on the television screen.

5) Iris I1 was positioned between mirrors M1 and M2, and irises I2 and I3 were positioned between the height-reducing mirrors and mirror M5. The dichroic beam splitter BS1, which transmits above 600 and reflects below this wavelength, was positioned at the entrance height and several inches in front of the AOM at approximately a 45^0 angle. The aperture of iris I1 was positioned to the height of the entrance aperture of the AOM, and the aperture of irises I2 and I3 were positioned three inches above the optical table. A beam height of three inches after the AOM was chosen since this was the center height of the mounted optical components that would be placed in the laser light path between the height-reducing mirrors and mirror M5.

6) The Helium-neon laser was activated. Using the support upon which the Helium-neon laser is situated, and the first two mirrors, M1 and M2, the beam of the laser light was directed through iris I1 and into the entrance aperture of the AOM. Iris I1 was closed down to allow only the collimated core of the laser beam to pass. Thus the laser beam of light from the Helium-neon laser entered the AOM

parallel to the surface of the optical table.

7) The AOM was activated. The output at the exit aperture of the AOM was checked to see if there were only two dominant orders present (indicating that the laser beam was entering the AOM at the Bragg angle). If more than two dominant laser beams were present, then mirrors M1 and M2 were adjusted for direction of the beam in the horizontal plane (i.e., the plane parallel to the surface of the table) until only two dominant orders of light were present.

8) The height-reducing mirrors, situated after the exit aperture of the AOM, were adjusted to direct the laser beams exiting the AOM through the aperture of irises I2 and I3. This resulted in the two laser beams being parallel to the surface of the optical table and three inches above the surface.

9) A 250-mm-focal-length cylindrical lens was secured to a support. This lens was then slid back and forth along the beam path until the two slits of light created by this lens had parallel beam paths. (Thus the cylindrical lens was now positioned such that the distance of travel by the light from the virtual diffraction origin of the AOM to the cylindrical lens was 250 mm.)

10) Irises I2 and I3 were closed down to the smallest diameter possible that still allowed for the transmission of both laser beams.

11) A 150-mm-focal-length achromat (AL1) was secured into threaded tubing. The tubing was inserted in tubing rings situated after the cylindrical lens. The achromat was slid back and forth until the image formed after the achromat were two slits of light in focus 150 mm beyond the achromat, and had parallel light in one dimension for each slit image. (As expected, the distance between the cylindrical lens and the achromat lens was 400 mm, the sum of the focal length of each lens.) The achromat lens AL1 was secured in place.

12) An iris (iris I4) was attached to the support structure (holding the camera, achromat, microscope objective, and long pass filter) at its bottom port. Note that the aperture of the iris was centered in the middle of the port and, hence, centered around the vertical axis of the support structure.

13) A filter that transmits the excitation wavelengths of 473 nm and 632 nm and reflects the fluorescence photons to the detection section of the setup (i.e., dichroic beam splitter BS2) was positioned at a 45^0 degree angle immediately after the achromat lens AL1.

14) The mirror underneath the sample stage and support structure (mirror M5) was put into place. The mirror was then adjusted such that the two laser beams struck the mirror symmetrically around its center, and were redirected through the aperture of iris I4. The television screen displaying the output of the CCD camera was viewed. If the laser light was perpendicularly striking the microscope slide,

the image of the reflection of the laser light off the top and bottom surface of the microscope slide would coincide with one another. If the two reflections did not coincide, then mirror M5 had to be readjusted for orientation and position. This adjustment of mirror M5 had to continue until the two laser beams were striking the center of mirror M5, traveling through the aperture of the iris, and perpendicularly striking the microscope slide.

15) The support structure, sample stage, and mirror M5 were then moved in unison towards or away from AL1 (and BS2) until the slits of light on the top surface of the microscope slide seen on the television screen interfaced with the CCD camera were in focus and had parallel light in one direction. (This meant that AL1 was 150 mm from the back focal plane of the microscope objective.) Note that usually step (14) had to be repeated for each new location of the support structure, sample stage, and mirror M5. Conversely, one could move AL1, the cylindrical lens, the height-reducing mirror, and the AOM in unison back and forth. However, this would also require the readjustment of mirrors M1 and M2.

16) An adjustable aperture was placed at the point of sharpest focus of the light slits between the cylindrical lens and the achromat AL1 (i.e., at the back focal plane of the cylindrical lens) to provide a means of controlling the length of the excitation slits at the test sample stage.

17) The achromats AL3 and AL4 in the detector section of the experimental setup, used to focus the fluorescence photons, were each secured into separate threaded tubing. The threaded tubings were slid into tubing rings. The tubing rings allowed for sliding of the tubings along the axial axis of the achromats.

18) A structure comprised of a photomultiplier tube (PMT), a mechanical slit 200 μ by 3 mm (situated in front of the PMT), and a three-axis stage was positioned behind each threaded tubing such that the axial center of the mechanical slit approximately coincided with the axial center of the achromat.

19) An iris was secured to the front and back of each threaded tubing. All the irises were closed down to small apertures.

20) The slits of light formed at the sample stage were partially reflected from the top surface of the microscope slide and directed by the microscope objective, mirror M5, and beam splitter BS2, to mirror M6. This mirror was positioned such that one of the slits of light struck the aperture of the front iris of the tubing directly in front of mirror M6 (i.e., the tubing containing achromat AL3), while the other slit of light struck mirror M7. The height of the tubing containing AL3 was adjusted and mirror M6 reoriented so that the slit of light striking the aperture of the front iris passed through the back iris as well. Mirror M7 was repositioned to intercept the other slit and direct it into the front iris of the tubing containing AL4. The height of the tubing containing achromat AL4 was adjusted and mirror M7 reoriented so that the slit of light directed at the tubing passed through the two

irises attached to the tubing. (This alignment was done to maximize the collection by the achromats of the fluorescence photons originating at the sample stage.)

21) The irises were removed from the threaded tubings. The tubings were slid back and forth until the sharpest image of the light slit from the sample stage directed through each tubing formed at the plane of each mechanical slit. The tubings were secured in place.

22) The mechanical slits were repositioned such that the light slit focused on them passed through the mechanical slit. Thus, each mechanical slit in the detection section was now confocal with their excitation slit of light formed at the sample stage.

23) A bandpass filter, used to reduce or eliminate light noise sources, was secured into each threaded tubing. The filter was placed in front of the achromat (i.e., the light entering the threaded tubing to be collected by the achromat and focused onto the mechanical slit first passed through the bandpass filter) to avoid compromising the focusing of the light by the achromat.

II) Two slits at wavelength 473 nm

If it was desired to create a double slit using the 473 nm laser rather than the 632 nm laser, then the above alignment procedure for two slits at 632 nm is executed with steps (5), (6), and (7) replaced with the following:

5) The beam expander and the beam blocker were positioned between the diode laser and mirror M3, and irises I2 and I3 were positioned between the height-reducing mirrors and mirror M5. The dichroic beam splitter BS1, which transmits above 600 and reflects below this wavelength, was positioned at the entrance height and several inches in front of the AOM at approximately a 45^0 angle. The aperture of the beam blocker was positioned to the height of the entrance aperture of the AOM, and the aperture of irises I2 and I3 were positioned three inches above the optical table. A beam height of three inches after the AOM was chosen since this was the center height of the mounted optical components that would be placed in the laser light path between the height-reducing mirrors and mirror M5.

6) The diode laser was activated. Using the support upon which the diode laser is situated, and the mirrors M3 and M4, the beam of the laser light was directed into the entrance aperture of the AOM. Thus the laser beam of light from the diode laser entered the AOM parallel to the surface of the optical table.

7) The AOM was activated. The output at the exit aperture of the AOM was checked to see if there were only two dominant orders present (indicating that the laser beam was entering the AOM at the Bragg angle). If more than two

dominant laser beams were present, then mirrors M3 and M4 were adjusted for direction of the beam in the horizontal plane (i.e., the plane parallel to the surface of the table) until only two dominant orders of light were present.

III) One slit at wavelength 632 nm and the other at 473 nm

If it is desired to create a two excitation slits with one at 473 nm and the other at 632 nm, then the above alignment procedure for two slits at 632 nm is executed with step (7) replaced with the following:

7) The beam expander and the beam blocker were positioned between the diode laser and mirror M3. The aperture of the beam blocker was positioned to the height of the entrance aperture of the AOM. The diode laser was activated. Using the support upon which the diode laser is situated, and the mirrors M3 and M4, the beam of the laser light was directed into the entrance aperture of the AOM. Thus the laser beam of light from the diode laser entered the AOM parallel to the surface of the optical table. (Note that in step (15), the beam expander may have to be adjusted to get the blue slit and red slit to focus at the same plane.)

Once the system was aligned, data acquisition using the device could begin. The data acquired from biological specimens using the two-slit, two-color system are discussed in the last chapter. The data acquired from the 473 nm channel using fluorescent beads are discussed below.

Molecules of Equivalent Soluble Fluorochrome (MESF) Tests

Beads 7.4 microns in diameter and labeled with the fluorochrome molecule fluorescein were acquired to test the detection sensitivity of the 473 nm channel. The required beads to test the detection sensitivity of the 632 nm channel were not immediately available. The beads acquired were microspheres that are used for calibration in standard flow cytometry, and are labeled with a known number of molecules. The MESF value of the bead specifies the number of fluorochrome molecules labeling the bead. The tests that were performed involved placing the fluorescent beads on a microscope slide, exciting the beads with the 473 nm laser pulsed, and detecting the fluorescence photons emitted by the beads. The 473 nm laser was pulsed by placing a chopper wheel in the beam path of the laser. The purpose of the pulsing was to simulate the excitation process that the labeled flowing cells in the blood stream of an animal

(from which data would be acquired) would experience.

The absorption/emission spectra of fluorescein is shown below. Maximum absorption by fluorescein occurs for a wavelength of 488 nm, and peak emission by

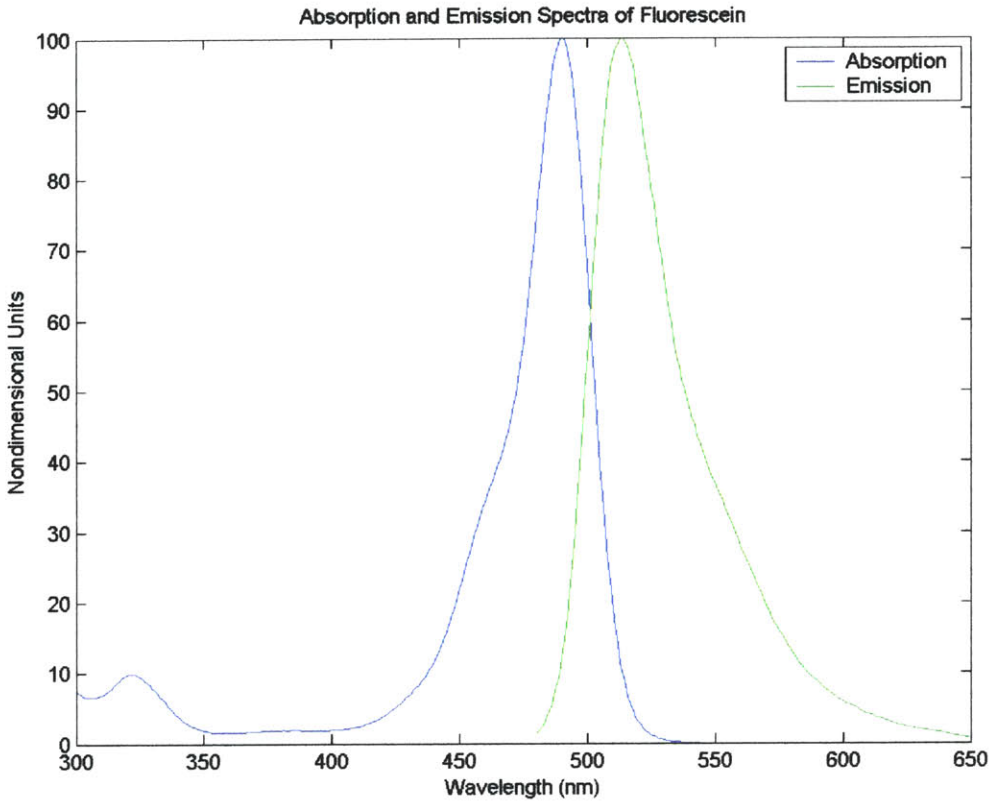

Figure 2.12 Absorption and Emission Spectra of Fluorescein

fluorescein occurs at 513 nm. Fluorescein was a good choice for the fluorochrome molecule to investigate the performance of the 473 nm channel because its absorption and emission spectra are similar to those of the GFP, as well as other fluorochrome molecules that were planned to be part of experiments to be performed using the cytometer.

The light pulse that was used to excite the beads is shown in Figure 2.13. As mentioned above, the pulse was created by a chopper wheel placed in the path of the

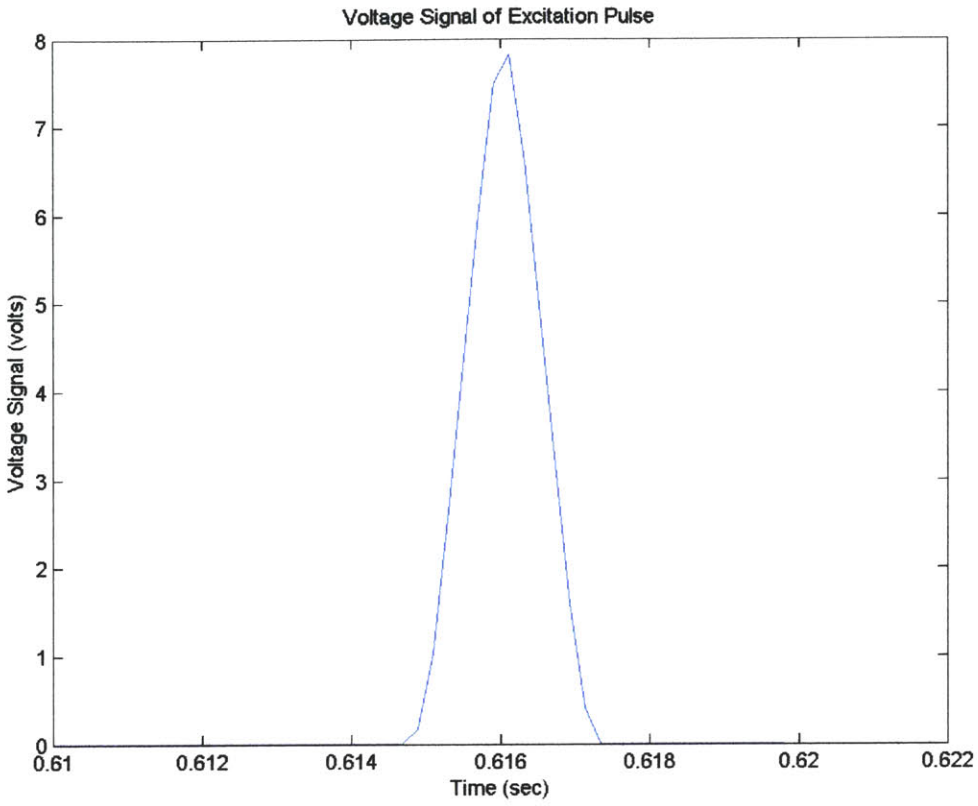

Figure 2.13 Excitation Pulse Shape

laser beam. The angular frequency of the chopper wheel was adjusted until the desired pulse width was obtained. This pulse width was used because it was representative of the exposure to the excitation light experienced by a cell 10 microns in diameter traveling at 10 mm/sec. This velocity is at the high end of the expected cell velocities for the experiments planned, so the results of system performance of the blue channel would be conservative.

The beads were prepared by diluting them in de-ionized water. This allowed for the isolation and excitation of a single bead. The beads were place on a microscope slide and placed on the sample stage. A quartz microscope slide was used to avoid autofluorescence background from the slide, although surface contaminants on the slide could still contribute to background noise.

The first beads from which data was acquired were beads that had an MESF value of 67451. (As mentioned earlier, the MESF rating is the number of molecules of the

fluorochrome being used to label the bead. Thus, for the beads being used in the experiment, one bead having an MESF value of 67451 fluoresces with the same intensity as a cell labeled with 67451 molecules of fluorescein.) The results obtained from the beads having an MESF value of 67451 are shown below in Figure 2.14, along with the background noise level present. The fluorescein labeled beads yielded voltage spikes 1.1 volts in height. (The acceleration voltage between the electron multiplying stages of the PMT was 750 volts.) The smaller spikes after 17.96

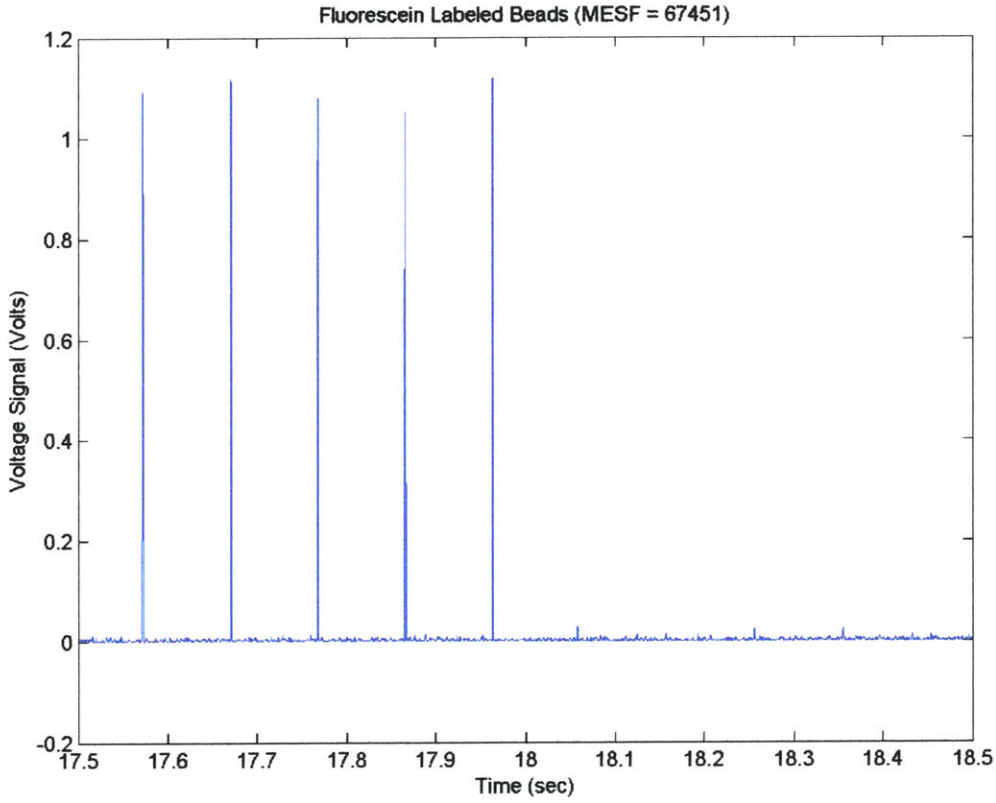

Figure 2.14 Voltage signal from fluorescein labeled beads having an MESF value of 67451, as well as background noise level

seconds is the autofluorescence background from the slide superimposed upon the noise from other sources, such as the environment and the detector. This noise measurement was obtained by sliding the excitation slit off the bead to a dry area on the slide where there were no beads or water. It had a value of 0.04 volts. The resulting signal-to-background ratio obtained was approximately 28. The voltage noise level between the

spikes, when the laser light was blocked by the chopper wheel (and, hence, there was no autofluorescence from the microscope slide), was approximately 0.005 volts. This yielded a signal-to-noise ratio of approximately 220. (The noise level present without autofluorescence from the slide was also measured by removing the slide and maintaining the pulsed laser beam. The results are shown in Figure 2.15. As expected, it matched the noise level present in between the spikes of the trace above, having a value of approximately 0.005 volts as well.)

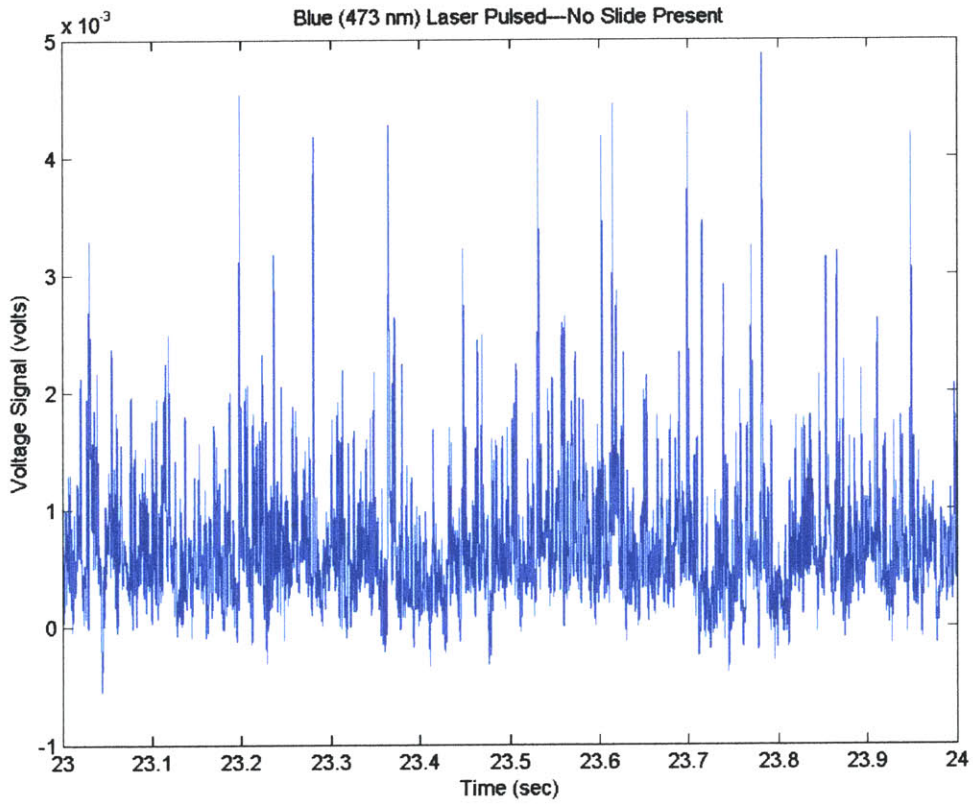

Figure 2.15 Noise level with blue (473 nm) laser pulsed and no slide present

Beads with an MESF value of 6288 were also used to check the performance of the blue channel. These beads had the lowest MESF value available for fluorescein (from the laboratory that supplied the beads). The data acquired is shown in Figure 2.16. The signal-to-noise ratio considering all noise sources was approximately 8, while for no autofluorescence present it was 70.

The results of these tests indicate that the present setup of the blue channel is of

sufficient sensitivity to detect circulating cells, provided that the labeled cells have at least a few thousand fluorochrome molecules attached. Other limiting factors in the detection of labeled circulating cells will be the level of autofluorescence background present from the skin of the animal, as well as the amount of scattering and absorption of the excitation and fluorescence light by the skin and blood of the animal.

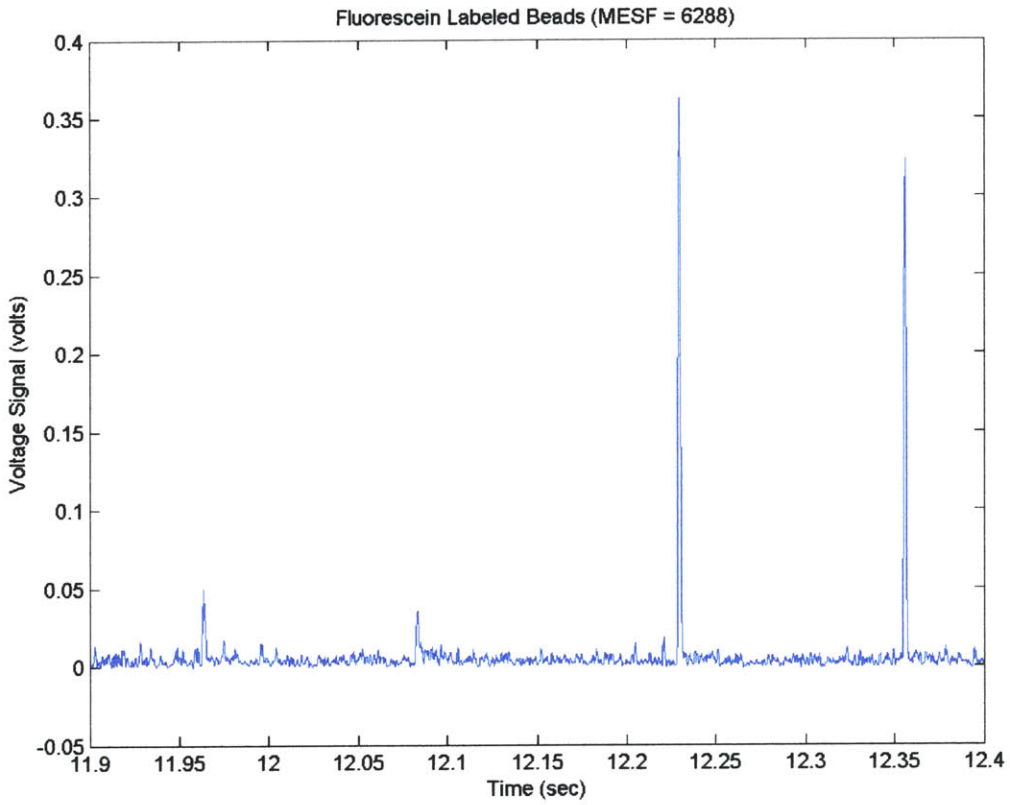

Figure 2.16 Voltage signal from fluorescein labeled beads having an MESF value of 6288, as well as background noise level

Chapter 3

Computer Codes for Analysis of Acquired Data

Introduction

The key purpose of analysis of the data is to determine how many cells have passed through the excitation laser lights of the in vivo flow cytometers. Analysis of the data acquired by the in vivo flow cytometers involves the usage of three computer codes, with a fourth one that can be used to display, in graphical format, the results of the analysis. The three codes used in the analysis of the acquired data are called binarysmoothingfile.m, binaryreadingcellcounting_new_John.m, and analysis8_newmod54_linear_MF_Scope.m. The first code listed, binarysmoothingfile.m, applies an averaging filter to the acquired data, and creates a file containing this averaged data. The number of points to be used in the averaging process is specified by the user. The next two codes, binaryreadingcellcounting_new_John.m, and analysis8_newmod54_linear_MF_Scope.m, are used to analyze the data for cell signal. Information such as cell count, cell signal peak voltage, and cell signal peak time (as well as other cell signal characteristics) are determined. The fourth code, countplot3_John.m, is used to display the results of the smoothing and analysis. All four codes are written in MATLAB. The three files used for data analysis are written as function files. This allows one to call and coordinate the execution of these files from a controlling program, as well transfer information between the codes. The controlling program is a graphical user interface program (GUI), and allows the user to select, at any one time, either the smoothing process (involving the code binarysmoothingfile.m) or the cell counting process (involving the codes binaryreadingcellcounting_new_John.m,, analysis8_newmod54_linear_MF_Scope.m), as well as the setting of the smoothing and counting parameters. The four codes, binarysmoothingfile.m, binaryreadingcellcounting_new_John.m,, analysis8_newmod54_linear_MF_Scope.m, and countplot3_John.m, are discussed below, and a step by step explanation of their execution is included. The GUI is also briefly discussed.

Binarysmoothingfile.m

The first code that is executed during data analysis is the file binarysmoothingfile.m. As mentioned above, the key purpose of this code is to smooth the acquired data (i.e., remove the high frequency components of the acquired data trace) and to store this smoothed data in a matrix and a dcf file. This code has as input parameters the filename (which includes the path of where to find the file) of the data to be smoothed, the filename and desired location of the binary dcf file which will contain the smoothed data (the data acquisition software Scope and DT Measure Foundry can only read specially formatted files called dcf files), the user specified start time and end time of the smoothing process, and the number of data points to use in the smoothing process.

The smoothing process of the acquired data is an averaging process. The user specifies, using the parameter nsm, the number of points to use to average each data point. The formula used in the averaging process is

$$data(q) = [data(q-nsm+1) + data(q-nsm+2) + \ldots + data(q-nsm+nsm)]/nsm$$

In other words, the value of the original point data(q) is replaced by the value obtained by the application of the above equation at point data(q). Note, that if there are less than nsm-1 points to the left of the point data(q), then the value of averaged data point will be equal to the summation of point data(q) and all the points available to the left of the point data(q), divided by nsm. Thus, the first point to be smoothed (call it data(1)) will have an averaged value of data(1)/nsm, the second point to be smoothed (call it data(2)) will have and averaged value of [data(1) + data(2)]/nsm, and so forth. Note that if nsm is 1, the averaged value of the data point is equal to the original value of the data point. The MATLAB filter which performs the averaging process has the form filter(ones(1,nsm)/nsm,1,data), where data is a matrix containing the points to be smoothed. (The matrix data, comprised of the points to be smoothed by reading the data from the dcf file containing the points to be smoothed, needs to be created because the filter function cannot acquire the data points directly from the dcf file.)

As mentioned above, the smoothed data is stored in a matrix and a file. The matrix is called data_smf and the file, which is a dcf file, is named by the user before the smoothing and analysis of the data begin. Note that even if one already has a smoothed

file from DT Measure Foundry, it is still required to execute the binarysmoothingfile.m code (with the parameter nsm equal to one) if the GUI is being used to guarantee that all the data files required to complete the data analysis process are properly named.

In addition to smoothing the acquired data, the code also checks that the start time and end time for the smoothing specified by the user are within the start time and end time available from the acquired data file. If the user specified start time is greater than the maximum start time available from the data file, execution of the code is terminated and the user is alerted. If the start time is valid but the end time is greater than that available from the data file, a new end time is calculated based upon the number of data points available beyond the start time data point. (In other words, the new end time calculated is the maximum end time available from the data file.)

The key steps executed, in chronological order, by the code binarysmoothingfile.m are:

1) The code receives as input the filename and location of the dcf file containing the data to be smoothed (data_filename), the filename and desired location of the dcf file that will contain the smoothed data (smdata_filename), the desired start time (starttime) and end time (endtime) of the smoothing process, and the number of points to use during the smoothing process (nsm). (line 1)

2) The dcf file into which the smoothed data will be saved is created and the matrix that will contain the smoothed data is cleared. (lines 21 to 22)

3) The dcf file containing the data to be smoothed (data_filename) is opened. (line 33)

4) The file ID (Magic) and file version (Version) are read from the dcf file that contains the data to be smoothed. The iteration counter i, which is the number of times the while loop, steps (5) to (14), has been executed in the present call to this code, is set to zero and the end-of-file indicator feof(fid) is set to zero. (lines 35 to 38)

5) The while loop is entered, and the end-of-file indicator is checked. The while loop will execute until the end-of-file indicator is equal to one. If it does equal one, code execution is transferred to step (15). (lines 39 to 43)

6) The iteration counter is increased by one. (line 44)

7) The length of the entire binary data field in bytes (A(i).Length), the number of bytes per data point (A(i).data_width), the data format as whether it is integer or float (A(i).data_format), and the time increment (A(i).TimeIncrement) that was used to collect the data (i.e., time interval between data points) are read. (lines 45 to 72)

8) The user specified start time is checked to see that it does not exceed the start time available from the data file. If it does, execution of the while loop is terminated and code execution is transferred to step (15). Otherwise, the user specified start time is stored in the parameter A(i).starttime and code execution continues to step (9). (lines 73 to 76)

9) The computer pointer is placed at the data point whose time location is equal to the start time specified by the user. (lines 77 to 88)

10) The number of points available from the present position of the pointer to the end of the file is determined, and the number of points needed to reach the user specified end time is calculated. If the number of points available from the file is less than the number of points needed to reach the user specified end time, a new end time is calculated and is equal to the maximum end time available from the data file. Otherwise, the user specified end time is unaltered. (lines 89 to 99)

11) Based upon the end time determined in step (10) and the user specified start time, the number of numerical entries to read from the dcf file data_filename is determined and is stored in the parameter A(i).num_entries_final. (lines 100 to 103)

12) The data is read from the dcf file data_filename and is stored in the vector A(i).data. (line 104)

13) The time vector corresponding to the data vector of step (12) is created. Uniformly spaced time points are determined knowing A(i).starttime, A(i)TimeIncrement, and A(i)num_entries_final. (line 105)

14) The end-of-file indicator feof(fid) is set equal to one. (line 106)

15) Execution of the while loop steps (5) to (14) is terminated and the dcf file data_filename is closed. (line 110)

16) The user specified start time is again checked that it is not greater than the maximum time available from the dcf file data_filename. If it is, the user is informed of this error and code execution is terminated. Otherwise, code execution proceeds to step (17). (lines 112 to 114)

17) The data read from the dcf file data_filename and stored in the vector A(i).data (ie., the data to be smoothed), is read into a matrix called data. (line 116)

18) The data is smoothed and stored in the matrix data_sm. (line 117)

19) The smoothed data is read into the output argument data_smf of the computer code (function file) binarysmoothingfile.m, as well as into the dcf file smdata_filename. (lines 119 to 133)

Figure 3.1 shows an unsmoothed raw data trace. The raw data was acquired at a sampling frequency of 25 kHz. Figure 3.2 shows the raw data after it has been smoothed (i.e., averaged) by the code binarysmoothingfile.m. The number of points used in the smoothing process was 25.

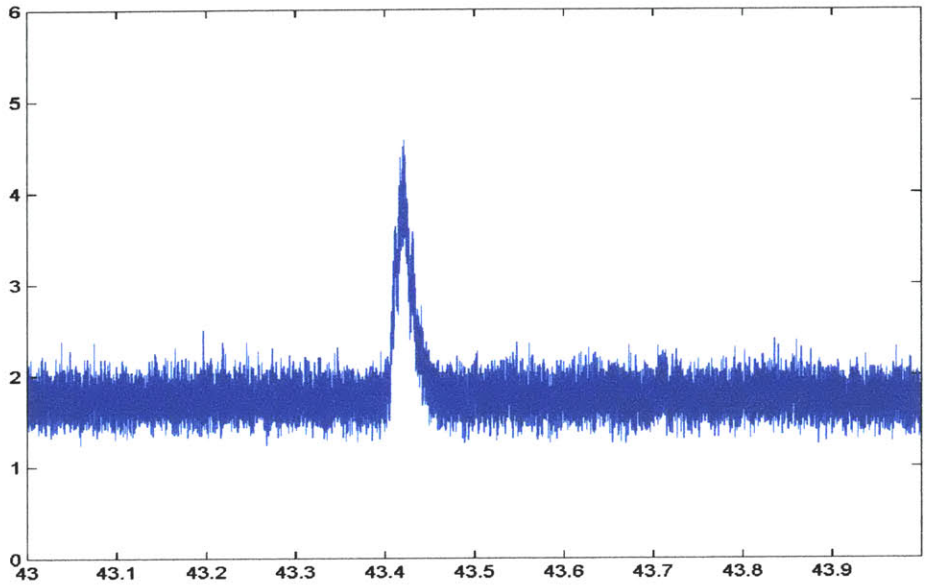

Figure 3.1 Raw (unsmoothed) data

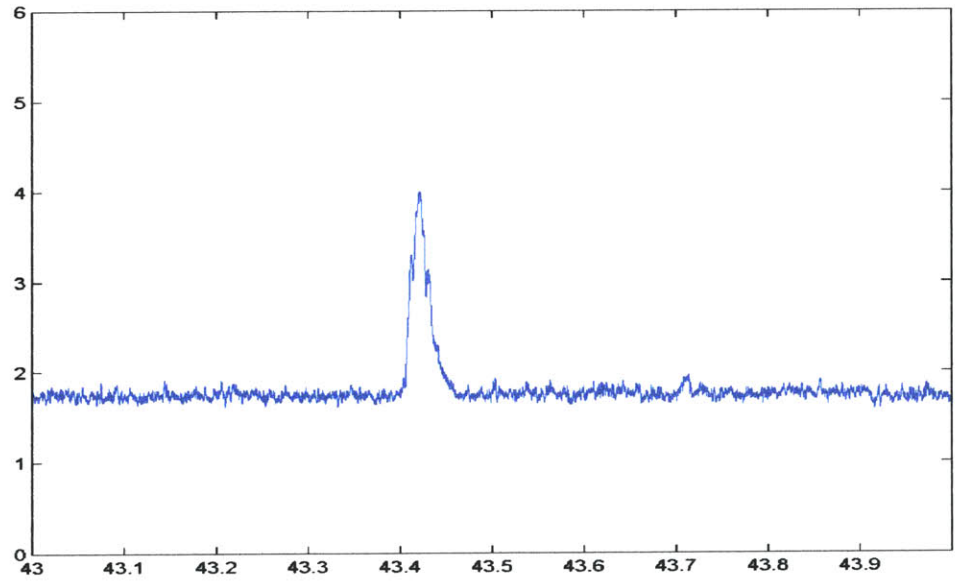

Figure 3.2 Averaged (smoothed) data

Binaryreadingcellcounting_new_John.m

The next code to be executed is binaryreadingcellcounting_new_John. As mentioned in the introduction, this code, along with the code analysis8_newmod54_linear_MF_Scope.m, is involved in cell signal analysis. This code performs several key functions. First, the code checks that the user-specified start time and end time for cell signal analysis are within the time boundaries of the smoothed file created by the code binarysmoothingfile.m. (Note that the user must specify a start time and end time both for the smoothing process and the cell signal analysis process since these processes are performed separately.) If the user-specified start time is less than that available from the smoothed data file, execution of the code is terminated and the user is informed. If the user-specified end time is greater than that available from the smoothed data file, the user is informed and a the maximum end time available from the file is made the new cell signal analysis end time. Second (if the user-specified start time was valid), the code reads the smoothed data from the file created by the code binarysmoothingfile.m, in user-specified increments, into a data matrix. Third, the code creates a time matrix that corresponds to the smoothed data read into the data matrix. Fourth, it sends to the code analysis8_newmod54_linear_MF_Scope.m the data matrix, time matrix, and several parameters needed for analysis of the data in the data matrix. Lastly, the code receives the data analysis information back from analysis8_newmod54_linear_MF_Scope.m, and stores the information in appropriate files for inspection and plotting (using countplot3_John) by the user. The key steps, in chronological order, performed by this code to accomplish the above are:

1) The code receives from the GUI:

filename---the file name and location of the smoothed data to be analyzed
root---the location and root name of the files that will be created by
binaryreadingcellcounting_new_John that will contain the analysis information
from the code analysis8_newmod54_linear_MF_Scope.m ,
starttime--- the user specified start time
endtime--- the user specified end time
nsm--- the number of points that were used during the data smoothing operation
nofint and sizeofint---product of these parameters is the minimum number of points to be
analyzed at one time (i.e., per iteration of the code
binaryreadingcellcounting_new_John) by

analysis8_newmod54_linear_MF_Scope.m,

vt---parameter used in by the data analysis code analysis8_newmod54_linear_MF_Scope.m in establishing the voltage boundary above which the data points are analyzed for cell signal and those below it are considered noise (and, hence, ignored for analysis),

stdv_th---parameter used by analysis8_newmod54_linear_MF_Scope.m in establishing the minimum height required for a voltage peak to be considered possible cell signal (both vt and stdv_th are discussed more fully in the analysis8_newmod54_linear_MF_Scope.m section)

counter_vector---vector whose entry values determine the method of calculation of the noise mean and standard deviation

rmst--- the user-specified parameter whose value is the voltage value below which all data points falling are used to calculate the noise statistics (used only if the entry read from counter_vector is one

slope---slope of the line drawn on the plot of height versus width (fwhm) for control data (see below)

intercept--- intercept of the line drawn on the plot of height versus width (fwhm) for control data (see below)

The vector counter_vector contains an integer or several integers between 1 and 6 inclusively, and the value of the entry read from this vector determines the method analysis8_newmod54_linear_MF_Scope.m uses to calculate the noise mean and standard deviation of the data sent to it by binaryreadingcellcounting_new_John. If the entry read has value of one, the parameter rmst is employed and is the voltage value below which all data points falling are used to calculate the noise statistics. If the entry read from the vector counter_vector has an integer value between 2 and 6 inclusively, then the noise statistics is determined by a percentage of the points irrelevant of their voltage value. For a value of 2, the lower (in voltage value) 50% of the points are used to determine the noise mean and standard deviation. For a value of 3,4,5, or 6, the lower 70%, 90%, 93%, and 65%, of the points are used, respectively. Concerning the parameters slope and intercept, these are determined by acquiring data from the selected data acquisition site on the animal without any labeled cells present in the animal's body. The data is analyzed with the slope and intercept set to zero in this code. The resulting peak heights (measured from a situation-dependent reference level determined by the peak counting code analysis8_newmod54_linear_MF_Scope.m), which are greater than the product of the parameter stdv_th and the standard deviation of the noise, and their corresponding full-width-half-maximum values, are plotted, and a line is drawn by the user specifying

that any peak below this line will be considered noise and will not be counted as cell signal (see Figure 3.3). The slope and intercept of this line drawn by the user is what is

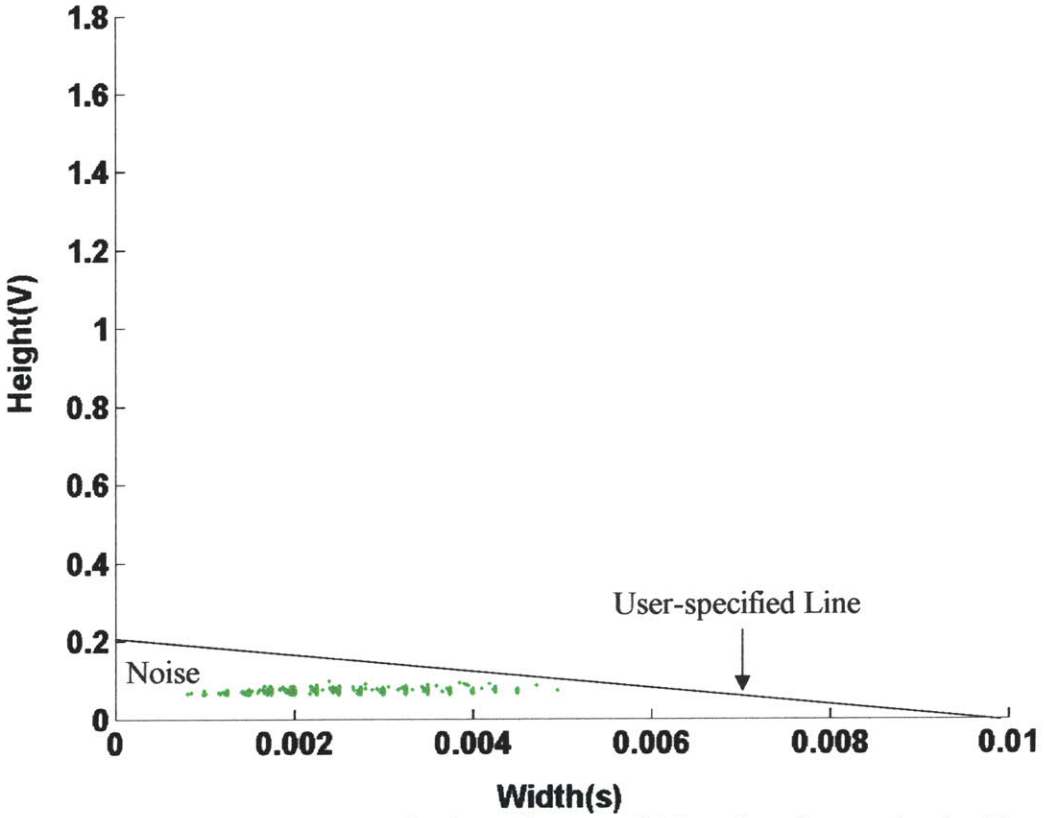

Figure 3.3 Noise are peaks from data acquisition site of test animal with no labeled cells present. Line is specified by the user above which is considered cell signal and below which is considered noise. This criterion is applied to peaks after satisfying minimum height criterion.

set equal to the parameters slope and intercept for cell signal analysis. Note that in order for the control data to be representative noise data during a test (where labeled cells are present and data is being acquired), all instrumentation settings should remain unchanged from control data acquisition to test data acquisition. (lines 1 to 2)

2) The code determines how many elements comprise the vector counter_vector. (line 35)

3) A for loop is entered (steps 3 to 27) where the argument of the for loop is called counter and whose initial value is set equal the value of the first element of

counter_vector . The end index of the for loop is set equal to the value of the last element of the vector counter_vector. (line 36)

4) The code sets to zero:

nvnewm---the up-to-date total cell peak number for counter_vector = 1

nvnewa50---the up-to-date total cell peak number for counter_vector = 2

nvnewa70---the up-to-date total cell peak number for counter_vector = 3

nvnewa90---the up-to-date total cell peak number for counter_vector = 4

nvnewa93---the up-to-date total cell peak number for counter_vector = 5

nvnewa95---the up-to-date total cell peak number for counter_vector = 6

diff---the difference between the total number of points that were sent to the code analysis8_newmod54_linear_MF_Scope.m and the total number of points that were actually analyzed for the previous iteration---this number will be nonzero for the present iteration if for the previous iteration the right end data point of the block of data sent to analysis8_newmod54_linear_MF_Scope.m was judged to be part of a potential cell signal (i.e., last data point was above the noise boundary determined by analysis8_newmod54_linear_MF_Scope.m)

nevaluated---the total number of data points analyzed by analysis8_newmod54_linear_MF_Scope.m thus far

EOFM---end-of-file (i.e., no more data points to be analyzed) marker

datamatrixcount---number of mounds of data above the noise boundary determined by analysis8_newmod54_linear_MF_Scope.m)

signaldatamatrixcount--- number of mounds of data above the noise boundary determined by analysis8_newmod54_linear_MF_Scope.m that have been judged by analysis8_newmod54_linear_MF_Scope.m to be cell signal

(lines 37 to 47)

5) The minimum total number of points to be analyzed for each iteration of binaryreadingcellcounting_new_John.m is calculated as the product of the user specified parameters sizeofint and nofint and is stored in the parameter m. The total number of

points analyzed for a particular iteration will be greater than this if the parameter diff for that iteration is nonzero (i.e., if for the previous iteration some of the points sent to the code analysis8_newmod54_linear_MF_Scope.m were not analyzed). (line 54)

6) The iteration counter j is set to zero. (line 59)

7) A while loop is entered (steps 7 to 25), and the end-of-file marker (EOFM) value is checked. The while loop will execute until the end-of-file marker is equal to one (EOFM will be set equal to one in analysis8_newmod54_linear_MF_Scope.m when the last data chunk has been set). If it does equal one, then evaluation of the value of EOFM at the beginning of the while loop will cause code execution to be transferred to step (26). (line 61)

8) The iteration counter j is increased in value by one, and the total number of points to be sent to analysis8_newmod54_linear_MF_Scope.m to be analyzed is calculated as the value of m + diff and stored in the parameter n. The dcf file filename containing the smoothed data is opened. (lines 62 to 63)

9) The file ID (Magic) and file version (Version) are read from this dcf file that contains the smoothed data. The iteration counter i, which is the number of times the while loop, steps (10) to (19), has been executed in the present call to this code, is set to zero and the end-of-file indicator (feof(fid)) is set to zero. (lines 72 to 73)

10) The while loop comprised of steps (10) to (19) is entered, and the end-of-file indicator feof(fid) is checked. The while loop will execute until the end-of-file indicator is equal to one. If it does equal one, code execution is transferred to step (20). (lines 74 to 78)

11) The iteration counter i is increased by one. (line 79)

12) The length of the entire binary data field in bytes (A(i).Length), the number of bytes per data point (A(i).data_width), the data format as whether it is integer or float (A(i).data_format), and the time increment (A(i).TimeIncrement) that was used to collect the data (i.e., time interval between data points) are read. (lines 80 to 108)

13) On the first iteration of the while loop comprised of steps (7) to (25) (i.e., for j equal to one), the user specified start time is checked to see that it does not exceed the start time available from the data file. If it does, the user is notified and execution of the code is terminated. Otherwise, the user specified start time is stored in the parameter A(i).starttime and code execution continues to step (14). If this is not the first iteration of the code (i.e., for j greater than or equal to two), the time of the first data point of the group of data points being analyzed is calculated (knowing the user specified start time, the time interval between data points, and the total number of points analyzed thus far by analysis8_newmod54_linear_MF_Scope.m) and is stored in the parameter A(i).starttime. (lines 109 to 112)

14) On the first iteration of the while loop comprised of steps (7) to (25), the number of points available from the user specified start time to the end of the file is determined, and the number of points needed to reach the user specified end time (from the user specified start time) is calculated. If the number of points available from the dcf file is less than the number of points needed to reach the user specified end time, a new end time is calculated and is equal to the maximum end time available from the data file. Otherwise, the user specified end time is unaltered. (lines 113 to 118)

15) The total number of points to be analyzed in the present iteration (i.e., n calculated in step (8)) is checked against the remaining number of points to be analyzed to reach the end time determined in step (14). If less than 200 points will remain to be analyzed after the present iteration of the code, then the parameter n is set equal to the total number of points remaining to be analyzed. Otherwise, n retains its value calculated in step (8) above. (lines 119 to 125)

16) The computer pointer is placed at the first data point of the group of points being analyzed in the present iteration of the code. (lines 128 to 140)

17) The data is read from the dcf file containing the smoothed data and is stored in the vector A(i).data. (lines 142 to 148)

18) The time vector corresponding to the data vector of step (17) is created. Uniformly spaced time points are determined knowing A(i).starttime, A(i).TimeIncrement, and n, and are stored in the vector A(i).time. (line 149)

19) The end-of-file indicator feof(fid) is set equal to one and, hence, execution of the while loop comprised of steps (10) to (19) is terminated. (line 150)

20) The dcf file containing the smoothed data is closed, and the information in A(i).data and A(i).time is read into the matrices data_smf and time, respectively. (lines 152 to 169)

21) The code sends to analysis8_newmod54_linear_MF_Scope.m the following parameters:

data_smf--data matrix containing the points to be analyzed
time--time matrix containing the time points of the data being analyzed
counter--parameter specifying how the noise mean and standard deviation will be calculated (i.e. what voltage value or what percentage of the data points will be used in the noise statistics calculations),
rmst-- the user specified voltage value below which any data point falling will be used in the calculation of the noise mean and standard deviation—this parameter is used only if counter has a value of one
vt--noise standard deviation multiplication factor used in establishing the voltage value below which is considered noise (and, hence, data points below this value are not analyzed) and above which is considered potential cell signal (and, hence, data points at or above this value are analyzed)
stdv_th-- noise standard deviation multiplication factor used in establishing the minimum height required for a voltage peak to be considered possible cell signal (to help discriminate a signal peak from a noise peak)
totalpoints--total number of data points (between start time and end time)
nevaluated--total number of data points already evaluated
nsm--number of points used during averaging (smoothing) process
slope--slope of the line drawn on the plot of height vs width of the noise spikes obtained

from the control data (see step (1) of binaryreadingcellcounting_new_John.m).
intercept--intercept of the line drawn on the plot of height vs width of the noise spikes obtained from the control data (see step(1) of binaryreadingcellcounting_new_John.m).

(line 199)

22) The code receives back from the waveform analysis code

analysis8_newmod54_linear_MF_Scope.m the following information:

rms---root-mean-square of noise
meanv---mean of noise
stdv---standard deviation of noise
n---number of data points of present iteration analyzed by the waveform analysis code
nv---number of cell signal peaks
index---index (i.e. point number) of cell signal peaks for group of data points of present iteration
t---time (absolute) of cell signal peaks
dt---time between cell signal peaks for present iteration
height---height (voltage value) of cell signal peaks measured from boundary between what is considered noise and what is considered potential cell signal
width---full-width-half-maximum of cell signal peaks
EOFM---end-of-data marker (if nonzero then all data points between start time and end time have been analyzed)
diff---number of data points of present iteration that were not analyzed due to being potential signal data at right time boundary (this will be zero if the last data point of data_smf is below the value of the parameter boundary or the data in data_smf is all of the remaining data to be analyzed of the dcf file)
nevaluated---total number of data points that have been evaluated for all iterations including the present iteration
datamatrixtotal---matrix containing all the data points (voltage values) that are above the noise boundary for present iteration
timematrixtotal---matrix containing all the time points of the data points that are above noise boundary the for present iteration
boundary---matrix containing boundary points for present iteration which separate noise from potential signal
firstnum---matrix containing the data points whose voltage value below which are calculated the noise parameters (mean and standard deviation) for the present iteration
absheight---matrix of data points that are the voltage value (absolute) of the peaks detected
signaldatamatrixtotal---matrix of data points (voltage values) that are cell signal for the present iteration
signaltimematrixtotal---matrix containing the time points of the cell signal points

(line 199)

23) After receiving this information from the waveform analysis code, binaryreadingcellcountin_new_John.m creates (if j equals one) or adds (if j is greater than or equal to two) the following matrices:

data_smftotal--matrix of all the data points (voltage values) that have been analyzed by the waveform analysis code analysis8_newmod54_linear_MF_Scope.m up to the present time
timetotal---matrixcontaining all the time points of the data points that have been analyzed by the waveform analysis code up to the present time
boundarytotal---matrix containing all the data points which form the boundary between noise and potential signal up to the present time
firstnumtotal---matrix containing all the data points up to the present time whose voltage value below which all data points falling have been used in the calculation of the noise mean and standard deviation
datamatrixtotaltotal---matrix containing all the data points up to the present time that are above the noise boundary
timematrixtotal---matrix containing all the time points of the data points up to the present time that are above the noise boundary

(lines 201 to 230)

24) If for the present iteration at least one cell peak is counted, then the code creates, if no cell peaks have been detected for any of the previous code iterations or this is the first iteration of the code, or modifies, if cell peaks have been detected for previous code iterations, the following matrices:

widtha95---matrix containing the full-width-half-maximum value of all the cell peaks counted up to the present time
heighta95---matrix containing the voltage height measured relative to a reference level determined by the waveform analysis code analysis8_newmod54_linear_MF_Scope.m of all the cell peaks counted up to the present time;
absheighta95---matrix containing the absolute height of all the cell peaks counted up to the present time
ta95---matrix containing time location of all the cell peaks counted up to the present time
dta95---matrix containing the time difference between the cell peaks within each iteration (i.e., each value of j)
indexa95---matrix containing the index (point number) of the cell peaks within each iteration
nvnewa95---matrix containing the total cell peak count up to the present time

signaldatamatrixcount--matrix containing the total number of signal mounds counted up to the present time

signaldatamatrixtotaltotal---matrix containing all the data points (absolute voltage values) up to the present time which are above boundary noise and are cell signal

signaltimematrixtotaltotal---matrix containing the time points of all the data points up to the present time which are above boundary noise and are cell signal

Note that the last three characters of the first seven matrices listed is contingent upon the counter value. Here the counter is assumed to have a value of six. If the counter value were different, the a95 ending would change to the following:

counter = 1---ending is m
counter = 2---ending is a50
counter = 3---ending is a70
counter = 4---ending is a90
counter = 5---ending is a93

(lines 232 to 304)

25) The while loop comprised of steps (7) to (25) continues until all data points between and including start time and end time have analyzed. On the last iteration (i.e., last execution of the while loop), the waveform analysis computer code analysis8_newmod54_linear_MF_Scope.m returns a nonzero value of the parameter EOFM to stop execution of this while loop.

26) After completion of analyzing all the data points between and including the start time and end time, the code creates the following matrices:

x1---matrix of numbers representing the mid value of the intervals into which the cell signal heights per minute will be separated

x2---matrix of numbers representing the mid value of the intervals into which the cell signal fwhm widths per minute will be separated

(lines 318 to 321)

27) The code checks the value of the for loop argument counter and directs code execution to the appropriate line so that the desired nomenclature can be attached to the

output matrices and file names created by this code. The desired nomenclature is as follows:

counter = 1---ending is _m
counter = 2---ending is _50
counter = 3---ending is _70
counter = 4---ending is _90
counter = 5---ending is _93
counter = 6---ending is _95

Assuming a counter value of six, if no cell peaks have been detected in any of the iterations, the user is alerted and only one more matrix called cellnum_95 is created. The matrix cellnum_95 contains the average number of cell signal peaks detected per minute. For the situation of no cell peaks detected, the matrix cellnum_95 will have zero as an entry. However, if at least one cell peak has been detected, the following matrices and files, in addition to the matrix cellnum_95, will be created:

histo_height_95---matrix containing histogram information of the cell signal heights per minute (histogram intervals are those of matrix x1)
histo_width_95---matrix containing histogram information of cell signal full-width-half-maximums per minute (histogram intervals are those of matrix x2)
root_95.txt---file containing the matrices heighta95, widtha95, ta95, dta95, indexa95
root_pkcta95.txt---file containing the matrices nvnewa95, cellnum_95
root_histo_height_95.txt---file containing the matrices x1, histo_height_95
root_histo_width_95.txt---file containing the matrices x2, histo_width_95
root_rawdataplot_95---file containg the matrix data_smftotal
root_boundarytotal---file containing the matrix boundarytotal
root_timedataplot_95---file containg the matrix timetotal
root_firstnumtotal---file containing the matrix firstnumtotal
root_moundplot_95---file containing the matrix datamatrixtotaltotal
root_timemoundplot_95---file containing the matrix timematrixtotaltotal
root_signalmoundplot_95---file containing the matrix signaldatamatrixtotaltotal
root_signaltimemoundplot_95---file containing the matrix signaltimematrixtotaltotal
root_peakdataplot_95---file containing the matrix absheighta95
root_peaktimeplot_95---file containing the matrix ta95

Note that the root term in the above file names is the location and root filename specified by the user in step (1) above.

(lines 323 to 645)

28) Code execution returns back to step (3) for the next value of counter. Steps (3) to (27) are repeated for all the values of the for loop argument counter.

analysis8_newmod54_linear_MF_Scope.m

The code analysis8_newmod54_linear_MF_Scope.m is the code that actually analyzes the data and determines if cell signal is present. The code receives from binaryreadingcellcounting_new_John.m the data matrix containing the data to be analyzed, the time matrix containing the time points of the data, and the value of the parameters counter, rmst, vt, stdv_th, totalpoints, nevaluated, nsm, slope and intercept (see execution step (19) of of binaryreadingcellcounting_new_John.m or step (1) below). The general approach of the code to the data analysis is as follows. First, the noise mean and standard deviation of the data are calculated based on the value of the parameter counter (see step (1) of of binaryreadingcellcounting_new_John.m). Second, a boundary voltage, equal to the mean of the noise plus the product of the parameter vt and the standard deviation of the noise, is calculated. Third, the number of data points above this boundary voltage is determined. (If there are none, then all the output arguments of this function file are set to zero and code execution is transferred back to binaryreadingcellcounting_new_John.m.) Fourth, the points above the boundary voltage are separated into mounds of data where the points of each mound are continuous in time (i.e., where any two successive data points have a time difference equal to the data sampling time interval used to acquire the data). Fifth, beginning with the first mound of data, each mound is analyzed to determine if it is cell signal or noise. If it is determined to be noise, then no further analysis of the mound is performed. If it is determined to be cell signal then the mound of data is further analyzed for the presence of multiple cell signal, and key features such as signal peak height, signal width, and time of signal peak height are determined. Sixth, once all the mounds of data have been analyzed, the information obtained is sent back to the code of binaryreadingcellcounting_new_John.m for storage into matrices and output files. The steps below delineate how the code analysis8_newmod54_linear_MF_Scope.m carries out the analysis of the data sent to it by the code binaryreadingcellcounting_new_John.m.

1) The waveform analysis code receives from binaryreadingcellcounting_new.m the

following matrices and parameters:

data--matrix containing the data points to be analyzed
time--matrix containing the time points of the data to be analyzed
counter--parameter specifying how the noise mean and standard deviation will be calculated (i.e. what voltage value or what percentage of the data points, starting at the lowest value, will be used in the noise statistics calculations),
rmst--the user specified voltage value below which any data point falling will be used in the calculation of the noise mean and standard deviation--this parameter is used only if counter has a value of one
vt--noise standard deviation multiplication factor used in establishing the voltage value below which is considered noise (and, hence, data points below this value are not analyzed) and above which is considered potential cell signal (and, hence, data points at or above this value are analyzed)
stdv_th-- noise standard deviation multiplication factor used in establishing the minimum height required for a voltage peak to be considered possible cell signal (to help discriminate a signal peak from a noise peak)
totalpoints--total number of data points (between start time and end time)
nevaluated--total number of data points already evaluated
nsm--number of points used during averaging (smoothing) process
slope--slope of line drawn on plot of height vs width of noise spikes obtained from control data (see step (1) of binaryreadingcellcounting_new_John.m) used to help discriminate noise from signal
intercept--intercept of line drawn on plot of height vs width of noise spikes obtained from control data (see step(1) of binaryreadingcellcounting_new_John.m) used to help discriminate noise from signal
(line 2)

2) The number of data points in the matrix data is determined. The matrix data_sm is created by copying into its first column the values of the matrix time, and into its second column the values of the matrix data. (lines 26 to 35)

3) The value of the parameter counter is read. If it is equal to one then all data points in the second column of the matrix data_sm that are below the value of the parameter rmst are located and their mean and standard deviation determined. Their mean value is stored in the parameter meanv and their standard deviation value is stored in the parameter stdv. These two parameters are taken to represent the mean and standard deviation of the noise present in the data. If the counter value is 2, 3,4,5, or 6 then the points are sorted from lowest voltage value to highest voltage value and, depending on the counter value, a

percentage of them (from the lowest voltage value) are used to determine the value of meanv and stdv. For the counter values 2, 3, 4, 5, and 6, the percentage of points used to determine the value of meanv and stdv is 50, 70, 90, 93, and 65, respectively. In addition, the maximum value of the data points used to determine the noise statistics is set equal to the parameter firstnum, which is the voltage value at or below all data points falling are included in the noise statistics calculation (lines 36 to 81). The red solid line of Figure 3.4 shows the firstnum voltage value for that voltage trace.

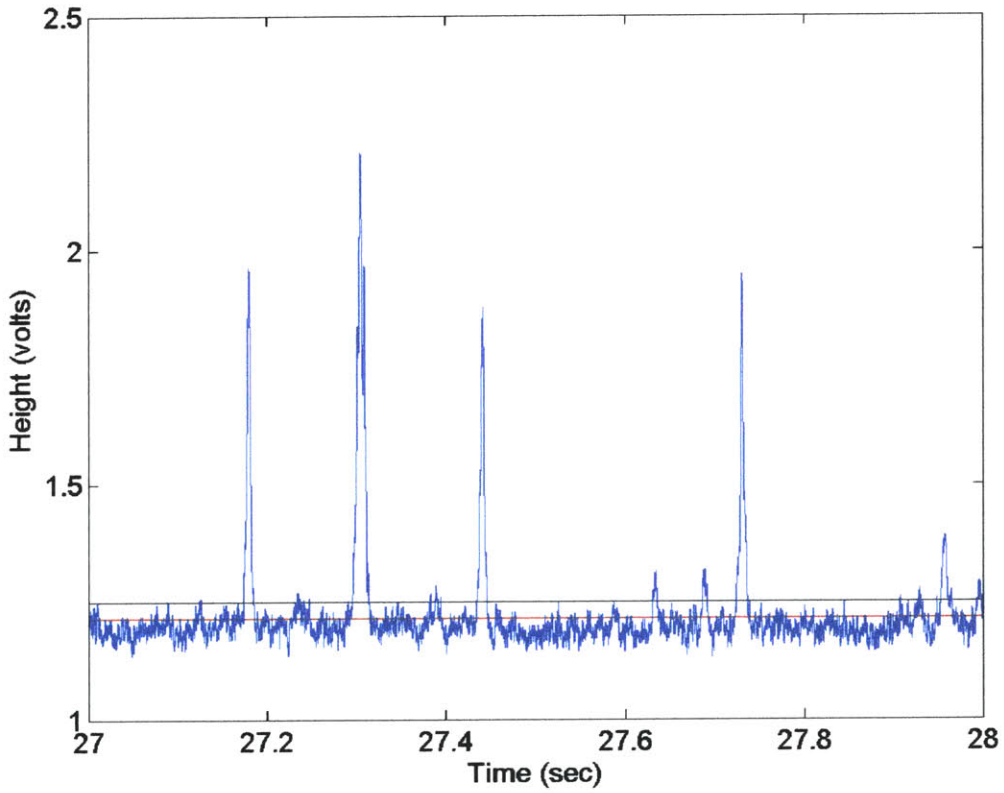

Figure 3.4 Voltage trace showing firstnum voltage value and boundary voltage value.

4) Using the mean and the standard deviation of the noise of the data, as well as the noise standard deviation multiplication factor (vt), the boundary voltage value at or below which is considered noise and above which is considered potential signal is calculated (line 83).

boundary = height (voltage value) of line below which is considered noise and above which is considered potential cell signal

= (mean of noise) + (vt)*(standard deviation of noise)
vt = noise standard deviation multiplication factor set by the user

The black solid line of Figure 3.4 shows the boundary voltage value for that voltage trace.

5) The waveform analysis code determines maximum number of data points of the data matrix that will be analyzed. The data will be analyzed up to the last point that is at or below the boundary voltage value calculated in step (4), unless the data matrix contains all the remaining data. If not all of the data points sent to analysis8_newmod54_linear_MF_Scope.m during this present iteration will be analyzed, the difference between the number sent to the code and the number that will be analyzed is calculated and assigned the variable name diff. At the end of execution of analysis8_newmod54_linear_MF_Scope.m for this present iteration, the variable diff will be sent back to binaryreadingcellcounting_new_John.m and the next iteration of points to be analyzed will include those not analyzed in the present iteration. Step (5) is performed to avoid counting a peak twice, which might occur if the peak appears at the right boundary of the data and continues into the next block of data points to be analyzed. (lines 84 to 100)

6) An end-of-file parameter (EOFM) is assigned one if all the remaining data points will be analyzed in this iteration. Otherwise its value remains zero. (lines 84 to 100)

7) If the number of points to be analyzed is not equal to zero (i.e., there are points at or below the boundary voltage value calculated in step (4), or all the remaining data points are being analyzed in this present iteration), the code continues to step (8). If it is equal to zero the code jumps to step (48). (line 106)

8) The index of the data points above the noise boundary (i.e., the boundary voltage calculated in step (4)) up to the last data point at or below the noise boundary are determined. If the matrix containing these indices has a length greater than one, then the code continues to step (9). If there are no points above the noise boundary then the code

jumps to step (48). (lines 107 to 109)

9) The data points that are above the noise boundary are determined. (line 111)

10) The difference in value between succeeding indices is calculated. (line 113)

11) The number of groups of data points that are above the noise boundary and are contiguous in time is determined. (In other words, the pools of data whose data points are all above the noise boundary level determined in step (4) and where the time separation between any two successive data points is the sampling time interval are determined.) These groups of data are referred to as signal mounds, mounds of data, or simply mounds. (lines 114 to 116)

12) A for loop is now entered. The steps (13) to (46) are executed for each mound of data. (line 119)

13) The index of the first data point (i.e., the data point with the lowest time value) and the index of the of the last data point (i.e., the data point with the highest time value) of the mound being analyzed is determined. (lines 120 to 136)

14) The data values and corresponding time values of the mound being analyzed are determined and stored into the matrices datamatrix and timematrix, respectively. (lines 141 to 143)

15) The matrix datamatrix is added to the matrix datamatrixtotal, which is a matrix that contains all the data points of all the mounds analyzed thus far for the present set of data points sent to this waveform analysis code. Similarly, the matrix timematrix is added to the matrix timematrixtotal, which is a matrix that contains all the time points of all the data points of all the mounds analyzed thus far for the present set of data points sent to this code. (lines 145 to 152)

16) The maximum (absolute) height of the mound and the corresponding index of the maximum height data point within the matrix datamatrix are determined. Also, the mound's halfheight, the halfway voltage value between the maximum height data point and the noise boundary, is calculated. (lines 154 to 155)

17) If the mound being analyzed is comprised of one data point, the full-width-half-maximum of the mound is set equal to zero. (lines 157 to 158)

18) If the mound being analyzed is comprised of more than one point, then the following steps (a) through (g) are executed: (lines 160 to 235)

a) For all the mound data points from the first one (index of one measured in datamatrix) to the maximum value data point (inclusive), the index of the data point that is closest in voltage value but does not exceed the halfheight of the mound and has no other data points that are lower in index and above it in voltage value is determined (i.e., the left halfheight index of mound is determined for the situation of the first data point of the mound having a voltage value less than the halfheight of the mound). If the first data point of the mound has a voltage value that is greater than or equal to the value of the halfheight of the mound, then the index of the left halfheight point of the mound is set equal to one. (lines 160 to 168)

b) If the halfheight of the mound is greater than the voltage value of at least one of the data points between the first data point and the maximum value data point (i.e., if the halfheight of the mound is greater in voltage value than the first data point of the mound), then by linear interpolation between the data point of the mound whose index is the left halfheight index of the mound, and the data point of the mound whose index is one greater than the left halfheight index of the mound (i.e., by linear interpolation between the two data points that bracket the halfheight of the mound for times that are less than the maximum value data point), the left halfheight time of the mound is determined. (lines169 to 172)

c) If the halfheight of the mound is less than or equal to the voltage value of all the data points between the first data point of the mound and the maximum value of the mound, then the following steps (i) to (iv) are executed: (lines 173 to 192)

(i) If the first data point of the mound is the first data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code (i.e., is the first data point in the matrix data_sm), and it is not the maximum value point of the mound (i.e., there are at least two points comprising the interval first data point to maximum value point of the mound inclusively), the indices of the points in this interval that have a voltage value greater than the voltage value of the first point of this interval are determined. (lines 174 to 177)

ii) If the first data point of the mound is the first data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code and there are no points in the interval first data point of the mound to the maximum value of the mound that are greater in voltage value than the first data point of this interval, (i.e., the first data point of the interval is the maximum value of the mound) then the left halfheight time of the mound is set equal to the time of the first data point of the mound. (line 178 to 179)

iii) If the first data point of the mound is the first data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code and there is at least one data point in the interval first data point to maximum value point (inclusive) which has a voltage value greater than the voltage value of the first data point of this interval, then the left halfheight time of the mound is determined by creating a spline between the first data point of this interval and the first data point of this interval which has a voltage value greater than the first data point of this interval and extrapolating to the halfheight voltage value of the mound. (lines 180 to 186)

iv) If the first data point of the mound is not the first data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code (and is also greater than or equal to the halfheight of the mound), then the left halfheight time of the mound is determined by interpolating in time between the data point preceeding the first data point of the mound and the first data point of the mound. (lines 187 to 192)

d) For all the mound data points from the first one after the maximum voltage value data point to the last data point of the mound (inclusive), the index of the data point that is closest in voltage value but does not exceed the halfheight of the mound and has no other

higher index data points above it in voltage value is determined (i.e., right halfheight index of mound is determined for the situation of the last data point of the mound having a voltage value less than the halfheight of the mound). If the last data point of the mound has a voltage value that is greater than or equal to the value of the halfheight of the mound, then the index of the right halfheight point of the mound is set equal to the index of the last data point of the mound. (lines 195 to 202)

e) If the halfheight of the mound is greater in value than the voltage value of at least one of the data points between the first data point after the maximum value data point and the last data point of the mound (i.e., if the halfheight of the mound is greater in voltage value than the last data point of the mound), then by linear interpolation between the data point of the mound whose index is the right halfheight index of the mound and the data point of the mound whose index is one less than the right halfheight index of the mound (i.e., by linear interpolation between the two data points that bracket the halfheight of the mound for times that are greater than the maximum value data point), the right halfheight time of the mound is determined. (lines 203 to 206)

f) If the halfheight of the mound is less than or equal to the voltage value of all the data points between the first data point after the maximum value data point and the last data point of the mound, then the following steps are executed: (lines 207 to 226)

i) If the last data point of the mound is the last data point of the present iteration data set sent by the code binaryreadingcellcounting_new.m code (i.e., the last data point of the matrix data_sm), and it is not the maximum value point of the mound (i.e., there are at least two points comprising the interval maximum value point of the mound to last data point of the mound inclusively) the indices of the points in this interval that have a voltage value greater than the voltage value of the last point of this interval are determined. (lines 207 to 211)

ii) If the last data point of the mound is the last data point of the present iteration data set sent by the code binaryreadingcellcounting_new.m code, and there are no points in the interval maximum value point of the mound to last data point of the mound (inclusive) that are greater in voltage value than the last data point of the interval, (i.e., the last data point of the mound is a repeat of the maximum value of the mound), then the right halfheight time of the mound is set equal to the time of

the last data point of the mound. (lines 212 to 213)

iii) If the last data point of the mound is the last data point of the present iteration data set sent by the code binaryreadingcellcounting_new.m code, and there is at least one data point in the interval maximum value point of the mound to last data point of the mound (inclusive) which has a voltage value greater than the voltage value of the last data point of this interval, then the right halfheight time of the mound is determined by creating a spline between the last data point which has a voltage value greater than the last data point of this interval and the last data point of this interval and extrapolating to the halfheight of the mound. (lines 214 to 220)

iv) If the last data point of the mound is not the last data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code (and is also greater than or equal to the voltage value of the halfheight of the mound), then the right halfheight time of the mound is determined by interpolating between the last data point of the mound and the first data point after the last data point of the mound. (lines 221 to 226)

g) After the left halfheight time and the right halfheight time of the mound have been determined, the full-width-half-maximum of the mound is determined. If the first data point of the mound is the first data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code, and it is also the maximum value of the mound, or the last data point of the mound is the last data point of the present iteration data sent by the code binaryreadingcellcounting_new.m code, and it is also the maximum value of the mound, then the full-width-half-maximum of the mound is set equal to twice the difference between the right halfheight time and the left halfheight time of the mound. For all other situations the full-width-half-maximum of the mound is calculated as the difference between the right halfheight time and the left halfheight time. (lines 229 to 235)

19) If the equations

(height of mound) - boundary - (slope)*(fullwidthhalfmaximum of mound) - intercept > 0

and

(height of mound) – boundary – (stdv_th) * (standard deviation of noise) > 0

where boundary = height (voltage value) of line below which is considered noise and above which is considered potential cell signal

= (mean of noise) + (vt)*(standard deviation of noise)

vt = noise standard deviation multiplication factor set by the user

stdv_th = noise standard deviation multiplication factor used in establishing the minimum height required for a voltage peak to be considered possible cell signal (to help discriminate a signal peak from a noise peak)

slope = slope of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where all the points below this line are considered noise

intercept = intercept of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where all the points below this line are considered noise

is satisfied, (i.e., if the point determined by the height of the mound and the fwhm of the mound is located above the two lines shown in Figure 3.5) then steps (20) through (44) are executed to further analyze the mound for multiple cell signals. If the equation is not satisfied, then the waveform analysis code execution jumps back to step (12), and the next signal mound is analyzed. (line 256)

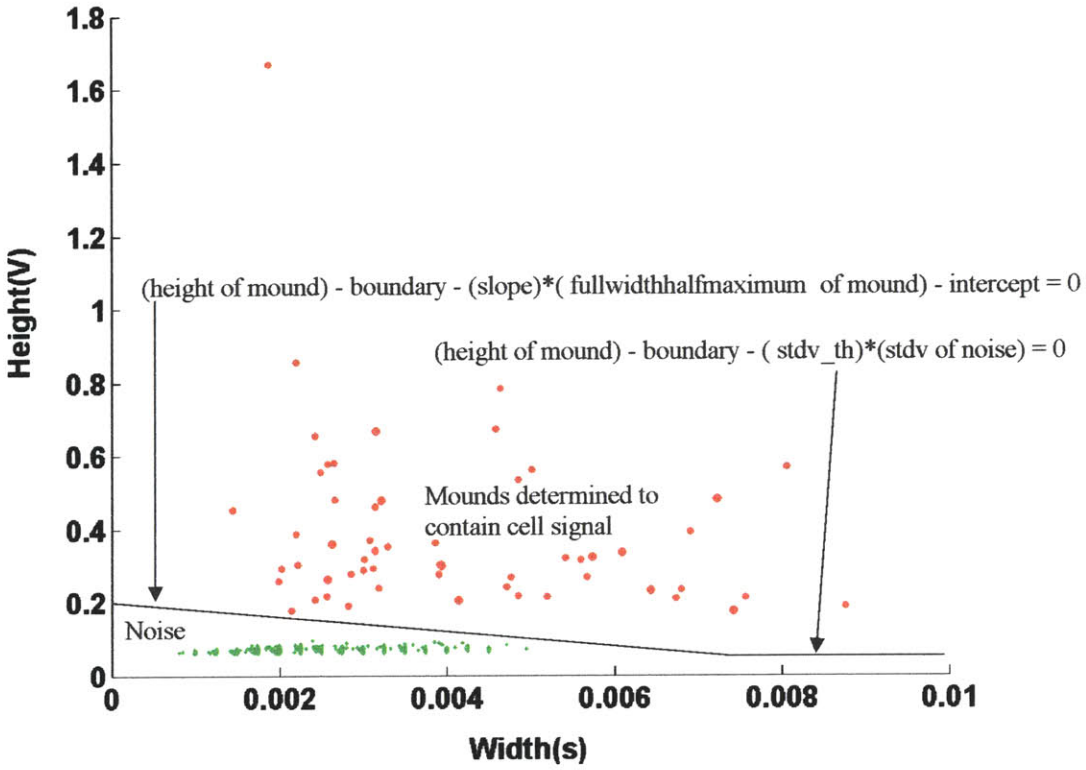

Figure 3.5 Separating noise from mounds determined to contain cell signal

20) The difference in height between adjacent data points [data point at time(k) - data point at time(k-1)] comprising the signal mound is calculated starting at k = 2 (second data point of the mound). (line 261)

21) The indices of the points of the signal mound being evaluated that have a lower voltage value than the previous point are determined. (lines 262 to 263)

22) Of the points that have a lower voltage value than the previous point, the indices of the points that also have the following point greater in voltage value are determined. In other words, the indices of the points which have the preceding point and the following point greater in voltage value are determined (i.e., the indices of the points which are relative minimum are determined). (lines 264 to 265)

23) A matrix called lastoneindices is created and is comprised of the first data point of the

signal mound, all the relative minimum points, and the last data point which has a preceding point that is greater in voltage value than itself (if it is not a relative minimum). These points are the critical points between which (inclusively) the data will be analyzed for the presence of a cell signal. If the signal mound is comprised of data points that are monotonically increasing or decreasing in voltage value (and, hence, there are no relative minimum points), then the first data point and last data point of the signal mound are made the critical points between which (inclusively) the data will be analyzed for the presence of a cell signal. (lines 267 to 290)

24) Next, a for loop is entered to analyze the signal mound. The for loop iterates from the second critical point to the last critical point of the signal mound one critical point at a time. Steps (25) to (43) are executed. (line 312)

25) The matrix minmatrix is formed, comprised of the indices (counted within the signal mound matrix) of the critical points from the last critical point where a signal was present (or if no signal has yet to be detected then from the first critical point) to the critical point before the present critical point that have a voltage value less than the voltage value of the present critical point. (line 479)

26) If the matrix minmatrix contains no elements, then the steps (27) to (37) are executed. If the length of minmatrix is nonzero, then code execution is transferred to step (38). (line 481)

27) The parameter bmax is set equal to b, which is the index (measured in the matrix lastoneindices) of the last critical point at which a signal was present or if no signal has yet to be detected then is equal to one. (line 482)

28) Two for loops are entered. The first for loop counter starts at b and iterates in increments of one to the index (measured in the matrix lastoneindices) of the critical point preceding the critical point which the for loop of step (24) is at presently. The second for loop counter starts at the critical point following the critical point at which the

first for loop is at and iterates in increments of one to the index (measured in the matrix lastoneindices) of the critical point which the for loop of step (24) is at presently. (lines 484 to 485)

29) If the first loop is at the index of a critical point which is less than bmax, then code execution is transferred back to step (28) to increase the value of the counter for the first for loop. Otherwise, code execution continues to step (30). (line 487)

30) The matrix datatobetested is created and is comprised of the data points of the signal mound from the critical point at which the first for loop is at of step (28) to the critical point at which the second for loop is at of step (28). The time points corresponding to these data points are determined and are stored in the matrix timetobetested. The maximum voltage value and the index (counted in the matrix datatobetested) of the maximum voltage value point are determined. The voltage value from which the peaks of the matrix datatobetested will be measured is the larger value of the two data endpoints of the matrix datatobetested and is set equal to the parameter boundary1. The voltage value halfway between the maximum voltage value of the matrix datatobetested and the parameter boundary1 from which the full-width-half-maximum of datatobetested will be calculated is determined and set equal to the parameter halfheight. (lines 489 to 493)

31) If the data set (i.e., datatobetested) being analyzed is comprised of one data point, the full-width-half-maximum of the data set is set equal to zero. (lines 515 to 516)

32) If the data set being analyzed is comprised of more than one data point, and the first data point of the data set is the first data point of the signal mound as well as the maximum point of the data set being analyzed, then the following steps (i) through (vi) are executed: (lines 518 to 543)

i) The half-height of the data set, called halfheightm, is calculated as the average voltage value between the first and last point of the data set. (line 519)

ii) The index (counted within the matrix datatobetested) of the data point that is the first data point to be below halfheightm and is greater in time value than the

maximum voltage point of the matrix datatobetested is determined and set equal to indexmindiffright. If no point can be found to have a voltage value below halfheightm and be greater in time than the maximum voltage point of the data set datatobetested, then indexmindiffright is set equal to the index of the data point with the largest time value (i.e., the last data point of the data set). (lines 520 to 527)

iii) If the point before the indexmindiffright point has the same voltage value as the indexmindiffright point, then the right half-height time of the data set datatobetested, called halfheighttimeright, is set equal to the time value of the data point whose index is indexmindiffright determined in step (ii) (which in this case will be the index of the last data point of the data set). If the point before the indexmindiffright point does not have the same voltage value as the indexmindiffright point, then the right half-height time of the data set is calculated by linear interpolation between the indexmindiffright point and the point before it. (lines 528 to 534)

iv) If the signal mound (of which the data set datatobetested is a part) starts at a data point which is not the first data point of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., not the first point of the matrix data_sm), then the left half-height time (i.e., the time that is less than the time of the maximum voltage point and has a voltage value that is equal to halfheightm), called halfheighttimeleft, is calculated by linear interpolation between the first data point of the data set (which is the maximum of the data set) and the point before it. The full-width-half-maximum of the data set is calculated by taking the difference between the right half-height time (halfheighttimeright) and the left half-height time (halfheighttimeleft). (lines 535 to 539)

v) If the signal mound (of which the data set is a part) starts at a data point which is the first data point of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., the first point of the signal mound is the first data point of the matrix data_sm), then the full-width-half-maximum of the data set is calculated by multiplying by two the difference between the right half-height time (halfheighttimeright) and the time value of the first data point of

the data set. (lines 540 to 542)

vi) The voltage value from which the peaks of the matrix datatobetested will be measured (i.e., boundary1) is set equal to the voltage value of the last data point of the data set. (line 543)

33) If the data set being analyzed (i.e., matrix datatobetested) is comprised of more than one data point, and the first data point of the data set being analyzed is the maximum point of the data set but is not the first data point of the signal mound, then the following steps are executed: (lines 545 to 565)

i) The half-height of the data set, called halfheightm, is calculated as the average voltage value between the first and last point of the data set. (line 546)

ii) The index (within the matrix datatobetested) of the data point that is the first data point to be below halfheightm and is greater in time value than the maximum voltage point of the data set is determined and set equal to indexmindiffright. If no point can be found to have a voltage value below halfheightm and be greater in time than the maximum voltage point of the data set, then indexmindiffright is set equal to the index of the data point with the largest time value (i.e., the last data point of the data set). (lines 547 to 554)

iii) If the point before the indexmindiffright point has the same voltage value as the indexmindiffright point, then the right half-height time of the data set, called halfheighttimeright, is set equal to the time value of the data point whose index is indexmindiffright determined in step (ii) (which in this case will be the index of the last data point of the data set). If the point before the indexmindiffright point does not have the same voltage value as the indexmindiffright point, then the right half-height time of the data set is calculated by linear interpolation between the indexmindiffright point and the point before it. (lines 555 to 561)

iv) The full-width-half-maximum of the data set is calculated to be the difference between the right half-height time (halfheighttimeright) and the time value of the first data point of the data set. (line 563)

v) The voltage value from which the peaks of the matrix datatobetested will be measured (i.e., boundary1) is set equal to the voltage value of the last data point

of the data set (i.e., the minimum voltage value point of the data set). (line 564)

34) If the data set being analyzed (i.e., datatobetested) is comprised of more than one data point, and the last data point of the data set is the last data point of the signal mound as well as the maximum point of the data set being analyzed, then the following steps (i) through (vi) are executed: (lines 568 to 593)

i) The half-height of the data set, called halfheightm, is calculated as the average voltage value between the first and last point of the data set. (line 569)

ii) The index (counted within the matrix datatobetested) of the data point that is the first data point to be below halfheightm and is smaller in time value than the maximum voltage point of the data set is determined and set equal to indexmindiffleft. If no point can be found to have a voltage value below halfheightm and be less in time than the maximum voltage point of the data set, then indexmindiffleft is set equal to the indexof the data point with the smallest time value (i.e., the first data point of the data set datatobetested). (lines 570 to 577)

iii) If the point after the indexmindiffleft point has the same voltage value as the indexmindiffleft point, then the left half-height time of the data set, called halfheighttimeleft, is set equal to the time value of the data point whose index is indexmindiffleft determined in step (ii) (which in this case will be the index of the first data point of the data set). If the point after the indexmindiffleft point does not have the same voltage value as the indexmindiffleft point, then the left half-height time of the data set is calculated by linear interpolation between the indexmindiffleft point and the point after it. (lines 578 to 584)

iv) If the signal mound (of which the data set is a part) ends at a data point which is not the last data point of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., not the last point of the matrix data_sm), then the right half-height time (i.e., the time that is greater than the time of the maximum voltage point and has a voltage value that is equal to halfheightm), called halfheighttimeright, is calculated by linear interpolation between the last data point of the data set and the point after it. The full-width-

half-maximum of the data set is calculated by taking the difference between the right half-height time (halfheighttimeright) and the left half-height time (halfheighttimeleft). (lines 585 to 589)

v) If the signal mound (of which the data set datatobetested is a part) ends at a data point which is the last data point of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., the last point of the signal mound is the last data point of the matrix data_sm), then the full-width-half-maximum of the data set is calculated by multiplying by two the difference between the left half-height time (halfheighttimeleft) and the time value of the last data point of the data set. (lines 590 to 592)

vi) The voltage value from which the peak of the matrix datatobetested is measured (i.e., boundary1) is set equal to the voltage value of the first data point of the data set. (line 593)

35) If the data set being analyzed is comprised of more than two data points, and the maximum point of the data set being analyzed is neither the first data point or the last data point of the data set datatobetested, then the following steps are executed: (lines 596 to 649)

i) The index (counted within the matrix datatobetested) of the data point that is the first data point to be below halfheight (halfheight is determined in step (30)) and is smaller in time value than the maximum voltage point of the data set is determined and set equal to indexmindiffleft. If no point can be found to have a voltage value below halfheight and be less in time than the maximum voltage point of the data set, then indexmindiffleft is set equal to the index of the data point with the smallest time value (i.e., the first data point of the data set). (lines 597 to 604)

ii) If the point after the indexmindiffleft point has the same voltage value as the indexmindiffleft point, then the left half-height time of the data set, called halfheighttimeleft, is set equal to the time value of the data point whose index is indexmindiffleft determined in step (i) (which in this case will be the index of the first data point of the data set). If the point after the indexmindiffleft point does

not have the same voltage value as the indexmindiffleft point, then the left half-height time of the data set is calculated by linear interpolation between the indexmindiffleft point and the point after it. (lines 605 to 611)

iii) The index (counted within the data set datatobetested) of the data point that is the first data point to be below halfheight and is greater in time value than the maximum voltage point of the data set is determined and set equal to indexmindiffright. If no point can be found to have a voltage value below halfheight and be greater in time than the maximum voltage point of the data set, then indexmindiffright is set equal to the index of the data point with the largest time value (i.e., the last data point of the data set). (lines 627 to 634)

iv) If the point before the indexmindiffright point has the same voltage value as the indexmindiffright point, then the right half-height time of the data set, called halfheighttimeright, is set equal to the time value of the data point whose index is indexmindiffright determined in step (iii) (which in this case will be the index of the last data point of the data set). If the point before the indexmindiffright point does not have the same voltage value as the indexmindiffright point, then the right half-height time of the data set is calculated by linear interpolation between the indexmindiffright point and the point before it. (lines 635 to 641)

v) the full-width-half-maximum of the data set is calculated to be the difference between the right half-height time (halfheighttimeright) and the left half-height time (halfheighttimeleft). (line 642)

vi) If the last data point of the data set being analyzed is the last data point of the signal mound (of which the data set being analyzed is a part), and the first data point of the data set has a voltage value that is less than the voltage value of the last data point of the data set, then the reference voltage level from which the data set peaks are measured (boundary1) is set equal to the voltage value of the first data point of the data set. (lines 643 to 645)

vii) If the first data point of the data set datatobetested being analyzed is the first data point of the signal mound (of which the data set being analyzed is a part), and the first data point of the data set has a voltage value that is greater than the voltage value of the last data point of the data set, then the reference voltage level

from which the data set peaks are measured (boundary1) is set equal to the voltage value of the last data point of the data set. (lines 646 to 648)

36) If the equations

(maxpoint) - boundary1 - (slope)*(fullwidthhalfmaximum1) - intercept > 0

and

(maxpoint) – boundary1 – (stdv_th) * (standard deviation of noise) > 0

where maxpoint = absolute voltage value of maximum voltage point of data set

boundary1 = height (voltage value) of line from which is measured the height of the data set (i.e., height of data set = maxpoint - boundary1)

slope = slope of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where all the points on the plot that fall below this line are considered noise—slope is determined by user

fullwidthhalfmaximum1 = width of the data set at the voltage value halfway between maxpoint and boundary1

intercept = intercept of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where all the points on the plot that fall below this line are considered noise—intercept is determined by user

stdv_th = noise standard deviation multiplication factor used in establishing the minimum height required for a voltage peak to be considered possible cell signal (to help discriminate a signal peak from a noise peak)

is satisfied (line 669), then it is concluded that a cell signal has been detected and steps (i) through (xii) below are executed: (lines 672 to 712)

i) The counter (signalnodecounter) for total number of cell signals of the signal mound being analyzed is increased by one.

ii) The counter (totalsignalnodecounter) for total number of cell signals of the

present iteration data block sent from binaryreadingcellcounting_new_John.m (i.e., of the matrix data_sm) is increased by one.

iii) The index (counted within the signal mound data) of the right endpoint (i.e., maximum value time point) of the data set being analyzed is stored in the variable signalnodeindice.

iv) The voltage value of the right endpoint (i.e., maximum value time point) of the data set datatobetested being analyzed is stored in the matrix signalnode.

v) The absolute voltage value of the peak voltage value of the data set being analyzed, maxpoint, is stored in the matrix absheights.

vi) The relative voltage value measured from boundary1 of the peak voltage value of the data set being analyzed is calculated and stored in the matrix heights.

vii) The time width from the first data point to the last data point of the data set being analyzed is calculated and stored in the matrix widths.

viii) The time of the peak of the data set being analyzed is stored in the matrix ts.

ix) The index (counted within the signal mound data) of the peak value of the data set being analyzed is stored in the matrix indexs.

x) The peak number (i.e., total number of cells detected thus far) for the present iteration data block sent from binaryreadingcellcounting_new_John.m (i.e., for the matrix data_sm) is determined and stored in the matrix nvs.

xi) bmax is set equal to the value of the counter of the second (i.e., inside) for loop of step (28), which is the index (measured in the matrix lastoneindices) of the relative minimum which is the right endpoint of this data set datatobetested determined to be a cell signal.

xii) Code execution jumps back to step (28).

If the equation above is not satisfied, then code execution immediately jumps back to step (28) (line 712)

37) Once both for loops of step (28) have been completed (i.e., once all critical points of the present signal mound have been analyzed) code execution jumps to step (24)

38) If the present critical point has a voltage value that is greater than the voltage value of at least one of the critical points from the last critical point where a signal was present (or if no signal has yet to be detected then from the first critical point) to the critical point before the present critical point (i.e., if the matrix minmatrix contains at least one element), then steps (39) through () are executed.

39) The index of all the critical points of the signal mound that precede in time the present critical point (i.e., the right endpoint of the present data set) and have a voltage value that is less than the voltage value of the present critical point is determined and stored in the matrix minmatrix1. (line 716)

40) The left index of the data set to be analyzed is chosen as the maximum index value between the right index of the last data set in which a signal was determined to be present (or if no signal has yet to be detected then from the first critical point) and the index of the last element of the matrix minmatrix1 (which is also a critical point). (line 717)

41) The matrix datatobetested is created and is comprised of the data points from the data point whose index is that determined in step (40) to the critical point at which the for loop is at of step (24). The time points corresponding to these data points are determined and are stored in the matrix timetobetested. The maximum voltage value and the index (in the matrix datatobetested) of the maximum voltage value point are determined. The voltage value from which the peaks of the matrix datatobetested will be measured is the larger value of the two data endpoints of the matrix datatobetested and is set equal to the parameter boundary1. The voltage value halfway between the maximum voltage value of the matrix datatobetested and the parameter boundary1 from which the full-width-half-maximum of datatobetested will be calculated is determined and set equal to the parameter halfheight. (lines 737 to 741)

42) Steps (31) to (35) are executed. (lines 743 to 877)

43) Once the maximum voltage value (maxpoint), the voltage value from which the peak

of the of the data set is being measured , and the full-width-half-maximum value (fullwidthhalfmaximum1) of the data set being analyzed has been determined, the equations

(maxpoint) - boundary1 - (slope)*(fullwidthhalfmaximum1) - intercept > 0

and

(maxpoint) – boundary1 – (stdv_th) * (standard deviation of noise) > 0

where

maxpoint = absolute voltage value of maximum voltage point of data set

boundary1 = height (voltage value) of line from which is measured the height of the data set (i.e., height of data set = maxpoint - boundary1)

slope = slope of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where most or all the points on the plot fall below this line--slope determined by user

fullwidthhalfmaximum1 = width of the data set at the voltage value halfway between maxpoint and boundary1

intercept = intercept of line drawn on plot of height vs width of control data (no labeled cells present in animal during data acquisition) where most or all the points on the plot fall below this line--intercept determined by user

stdv_th = noise standard deviation multiplication factor used in establishing the minimum height required for a voltage peak to be considered possible cell signal (to help discriminate a signal peak from a noise peak)

is applied (line 899). If it is satisfied, then steps (i) through (xii) below are executed: (lines 900 to 929)

i) The counter (signalnodecounter) for total number of cell signals of the signal mound being analyzed is increased.

ii) The counter (totalsignalnodecounter) for total number of cell signals of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., of the matrix data_sm) is increased by one

iii) The index (counted within the signal mound data) of the right endpoint (i.e., maximum value time point) of the data set being analyzed is stored in the variable

signalnodeindice

iv) The voltage value of the right endpoint (i.e., maximum value time point) of the data set being analyzed is stored in the matrix signalnode

v) The absolute voltage value of the peak voltage value of the data set being analyzed, maxpoint, is stored in the matrix absheights

vi) The relative voltage value measured from boundary1 of the peak voltage value of the data set being analyzed is calculated and stored in the matrix heights

vii) The time width from the first data point to the last data point of the data set being analyzed is calculated and stored in the matrix widths

viii) The time of the peak of the data set being analyzed is stored in the matrix ts

ix) The index (counted within the signal mound data) of the peak value of the data set being analyzed is stored in the matrix indexs

x) The peak number for the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., for the matrix data_sm) of the peak of the data set being analyzed is determined and stored in the matrix nvs

xi) The parameter b, which is the starting index of the first for loop (i.e., outside loop) of step (28), is set equal to the value of the counter of the for loop of step (24), which is the index (measured in the matrix lastoneindices) of the relative minimum which is the right endpoint of this data set determined to be a cell signal

xii) Code execution jumps back to step (24)

If the equation is not satisfied, then code execution immediately jumps back to step (24).

44) Once all the critical points of the signal mound have been exmamined, if no data set was judged to be a cell signal, then the entire mound of data, which has already been judged to be a signal mound in step (19), is concluded to be a cell signal and the following steps are executed: (lines 933 to 944)

i) The counter (totalsignalnodecounter) for total number of cell signals of the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., of the matrix data_sm) is increased by one.

ii) The index (counted within the signal mound data) of the right endpoint (i.e.,

maximum value time point) of the signal mound is stored in the variable signalnodeindice.

iii) The absolute voltage value of the peak voltage value of the signal mound being analyzed, maxpoint, is determined and is stored in the matrix absheights.

iv) The relative voltage value measured from boundary of the peak voltage value of the signal mound being analyzed is calculated and stored in the matrix heights.

v) The time width from the first data point to the last data point of the signal mound being analyzed is calculated and stored in the matrix widths.

vi) The time of the peak of the signal mound being analyzed is stored in the matrix ts.

vii) The index (counted within the signal mound data) of the peak value of the signal mound being analyzed is stored in the matrix indexs.

viii) The peak number for the present iteration data sent from binaryreadingcellcounting_new_John.m (i.e., for the matrix data_sm) of the peak of the signal mound being analyzed is determined and stored in the matrix nvs.

45) Once all the critical points of the signal mound have been exmamined, the data points comprising the signal mound and the corresponding time points are stored in the matrices signaldatamatrixtotal and signaltimematrixtotal, respectively. (lines 956 to 962)

46) Code execution is transferred back to step (13) and the next signal mound is analyzed.

47) Once all the signal mounds of data have been analyzed, the matrices absheights, heights, widths, ts, indexs, nvs, datamatrix, timematrixtotal, signaldatamatrixtotal, signaltimematrixtotal are transferred back to binaryreadingcellcounting_new_John.

48) If there are no data points above the noise boundary determined is step (4), then all the matrices listed in step (47) are set equal to zero and are transferred back to binaryreadingcellcounting_new_John. (lines 1028 to 1045)

Graphical User Interface (GUI)

The graphical user interface (GUI) is an umbrella code that can activate either the binarysmoothingfile.m code or the binaryreadingcellcounting_new_John.m code. The GUI allows one to select, from the keyboard and a displayed window, either the data smoothing process involving the code binarysmoothingfile.m, or the cell counting process involving the code binaryreadingcellcounting_new_John.m (which will call the data analysis code analysis8_newmod54_linear_MF_Scope.m). In addition, the GUI allows one to set (from the keyboard) all the user-specified parameters needed for the process selected, as well as the files to which the process is to be applied. More than one file at a time can be selected for smoothing or cell counting, although, as alluded to above, only one process can be selected at a time. The user-specified parameters that can be set using the GUI are the start time and end time of the smoothing process, the number of data points to use during the smoothing process, the start time and end time of the cell counting process, the number of data points to analyze per iteration of the code binaryreadingcellcounting_new_John.m (determined by the product of the parameters nofint and sizeofint), and the value of rmst, vt, stdv_th, slope, intercept, and counter. The GUI will also plot, using MATLAB, the smoothed data (voltage versus data point number) as well as the height versus width (full-width-half-maximum) of the cell signals counted.

The height versus width plot created allows the user to draw lines on the plot. This line drawing capability of the MATLAB plots is what is used to determine the value of the parameters slope and intercept. As discussed earlier and reiterated here, data is obtained with no labeled cells present in the animal's body from the data acquisition site to be used when labeled cells are present. This data is analyzed with the parameters slope and intercept set to zero. A plot of height vs width of the peaks from the control data is obtained from the GUI. The user draws a line on the plot where all peaks falling below this user-drawn line are considered noise. The slope and intercept of this line is what is entered as the value of slope and intercept when the data acquired with labeled cells is analyzed. (As discussed in the previous section, these slope and intercept numbers become part of a criterion in analysis8_newmod54_linear_MF_Scope.m used to

discriminate noise peaks from cell signal peaks.)

countplot3_John

The code countplot3_John.m is used to display graphically the smoothed data from the code binarysmoothingfile.m, as well as the results of the data analysis code analysis8_newmod54_linear_MF_Scope.m. The code will plot, in different colors, the smoothed data, the line called firstnum (below which all data points falling are used in the calculation of the noise mean and standard deviation), the line called boundary (above which are analyzed the data points for cell signal and below which is assumed noise data), the data that was above the line boundary and determined to be cell signal, the data that was above the line boundary and determined to be noise, and the peaks of the cell signals. In addition, the code allows the user to specify (from the keyboard) a start time for the plotting and checks that the start time is within the time available from the smoothed data file (i.e., the file that was analyzed). If the user-specified start time exceeds the maximum time available from the smoothed data file, the user is informed and is told what is the maximum time available from the file, and is asked to re-enter a new start time. Conversely, if the user-specified start time is less than the minimum time available from the smoothed data file, the user is informed and is told what is the minimum time available from the file, and is asked to re-enter a new start time. This process of asking the user for a start time continues until a valid start time is entered by the user. A time interval length (i.e., time width of the plot) is also asked of the user. If the user-specified time interval length is less than the time increment (i.e., spacing) of the data, then a time interval length of 0.2 seconds is assumed. (If a previous iteration with a valid time interval length has already taken place, then the user can just hit enter to retain the same time interval as the previous iteration.) A plot is created with the beginning time and ending time of the plot displayed in the MATLAB command window. This cycle of asking the user for a time interval length and then displaying the plot (as well as the beginning time and ending time of the plot), continues until all the data after the start time has been plotted, or until the user presses control-C. The steps below explain explicitly how the above tasks are performed by the code.

1) The user must open the code and specify the path of where to find the files and the root name of the files that will be plotted by the code. The files, which are created by the code binaryreadingcellcounting_new_John.m, are:

a. root_counted_rawdataplot_95—file contains the voltage values of the data that have been analyzed
b. root_counted_timedataplot_95—file contains the time points of the data that have been analyzed
c. root_counted_boundarytotal—file contains the voltage values of the noise boundary below which is considered noise and above which is considered potential signal
d. root_counted_firstnumtotal—file contains the voltage values of the boundary below which the data points are used to calculate the noise mean and standard deviation
e. root_counted_moundplot_95—file contains the data points above the noise boundary
f. root_counted_timemoundplot_95—file contains the time points of the data points above the noise boundary
g. root_counted_signalmoundplot_95—file contains the data points determined to be cell signal
h. root_counted_signaltimemoundplot_95—file contains the time points of the data points determined to be cell signal
i. root_counted_peakdataplot_95—file contains the data points determined to be a cell peak
j. root_counted_peaktimeplot_95—file contains the time points of the data points determined to be a cell peak

2) The code asks the user to enter (from the keyboard) a start time to begin the plotting of the data. (line 1)

3) The file root_counted_timedataplot_95 is opened and all the time data are read into the matrix N. (lines 3 to 4)

4) The maximum time, minimum time, and length of the matrix N are determined, as well as the time increment (i.e., time difference) between the time points in the matrix N. (lines 7 to 10)

5) The starting time specified by the user is checked against the minimum time and maximum time of the matrix N. If the starting time specified by the user is less than the minimum time of the matrix N, the user is warned of this fact and is told the minimum time available from the matrix N. Conversely, if the starting time specified by the user is greater than maximum time of the matrix N, the user is warned of this fact and is told the maximum time available from the matrix N. The user is then asked to enter a new start time. This step is repeated until the user specifies a start time that is greater than or equal to the minimum time of the matrix N, and less than or equal to the maximum time of the matrix N. (lines 13 to 24)

6) The index of the time points within the matrix N that are greater than or equal to the start time specified by the user are determined. A new matrix N is created which contains only those time points that are greater than or equal to the starting time specified by the user. The length of this new matrix N is determined. (lines 28 to 29)

7) The file root_counted_rawdataplot_95 is opened and all the data points are read into the matrix M. (lines 32 to 33)

8) A new matrix M is created which contains only those data points whose time values are greater than or equal to the user specified start time (i.e., only those data points whose corresponding time points are contained within the matrix N determined in step (6). (line 38)

9) The file root_counted__boundarytotal is opened and all the data points are read into the matrix P. (lines 40 to 41)

10) A new matrix P is created which contains only those data points whose time values are greater than or equal to the user specified start time (i.e., only those data points whose corresponding time points are contained within the matrix N determined in step (6). (line 44)

11) The file root_counted__firstnumtotal is opened and all the data points are read into the matrix F. (lines 46 to 47)

12) A new matrix F is created which contains only those data points whose time values are greater than or equal to the user specified start time (i.e., only those data points whose corresponding time points are contained within the matrix N determined in step (6). (line 50)

13) The user is asked to specify a time interval length (i.e., the time width of the plot to be displayed). If the time interval length entered by the user is less than the time increment of the data determined in step (4), then a time interval length of 0.2 seconds is assumed. (If a previous iteration for the present execution of this code has already taken place, the user can just hit enter and the previous time interval length will be used.) The time interval length is divided by the time increment of the data to determine the total number of time points required for the plot to be displayed. This number is stored in the parameter n. If the number of time points required for the plot to be displayed (i.e., n) plus the number of time points already used for all past iterations (i.e., for all past plots for the present execution of the code) plus 200 is greater than the total number of time points of the matrix N determined in step (6), then n is set equal to the number of time points remaining to be plotted. Otherwise n is left unchanged. (lines 52 to 74)

14) If there is a plot from a previous iteration being displayed, it is cleared. (line 76)

15) A new matrix NN is created which contains the entries of the matrix N from the first time point of the present iteration to the last time point of the present iteration. The matrices MM, PP, and FF are created from the matrices M, P, and F, respectively, the same way. Thus, one now has matrices that contain the present iteration time values (matrix NN), the present iteration voltage (i.e., boundary) values which separate noise from potential signal (matrix PP), the present iteration voltage (i.e., firstnum) values below which any data point falling is used in the noise statistics calculations (matrix FF),

and the present iteration voltage values of the smoothed data analyzed (matrix MM). (lines 78 to 100)

16) The code also creates matrices that, for the present time interval being inspected, contain the data points that are above the boundary voltage values (matrix T), the time points corresponding to the data points that are above the boundary voltage values (matrix S), the data points that are above the boundary voltage values and have been judged to be cell signal (matrix V), the time points corresponding to the data points that are above the boundary voltage values and have been judged to be cell signal (matrix U), the data points that have been judged to be the peak value of cell signal (matrix Q), and the time points of the data points that have been judged to be the peak value of cell signal (matrix R). (lines 103 to 143)

17) Matrices PP, FF, and MM are all plotted against matrix NN on the same plot, each using a different color. In addition, on this same plot and again in unique colors, matrix T is plotted against matrix S, matrix V is plotted against matrix U, and matrix Q is plotted against matrix R. Thus, one has a visual display that sums up succinctly and clearly all the results of the analysis of the data from the in vivo flow cytometer. In addition, the start time and end time of the plot is displayed in the MATLAB command window. (lines 148 to 168)

18) The parameter n, the total number of time points of the matrix N (created in step (6) above) that were plotted in this iteration (i.e., the time points of matrix NN), is added to the total number of time points of this matrix plotted in all previous iterations (for the present execution of the code). (line170)
19) Steps (13) to (18) are repeated until all the time points of the matrix N created in step (6) have been plotted, or until the user terminates the process with a control-C.

The plot of Figure 3.6 is that created by countplot3_John. The red solid line is the firstnum line below which all data falling was used in the calculation of the noise statistics (matrix FF). The black solid line is the noise boundary line above which is

considered potential cell signal and below which is considered noise (matrix PP). The blue part of the trace is the data below the noise boundary (from matrix MM). The black part of the trace is the data above the noise boundary line which is considered noise (from matrix T). The yellow part of the trace is the data above the noise boundary that is considered cell signal (matrix V). The red points are the peak values of each cell signal (matrix Q).

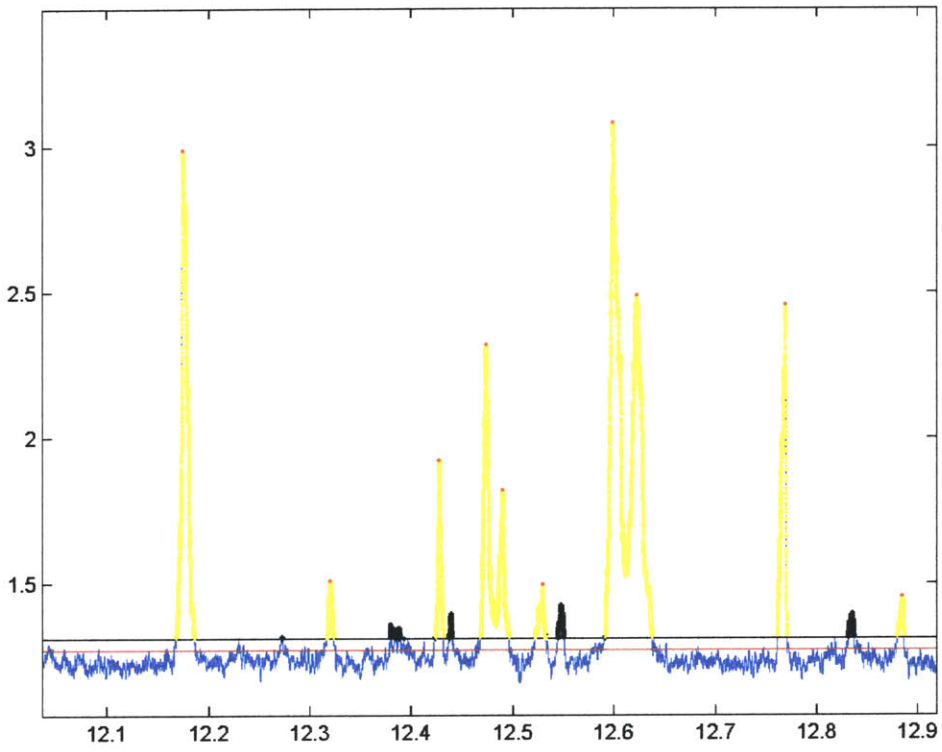

Figure 3.6 Plot from countplot3_John

Chapter 4

Application of In Vivo Flow Cytometry and Outlook

Introduction

Both in vivo flow cytometers are fully functional. Both the single-slit, single-color and two-slit, two-color in vivo flow cytometers have been and are presently being used to acquire data involving circulating labeled cells in the bloodstream of animals. The labeled cells have been ex vivo labeled cells (i.e., the cells were labeled outside the body of the animal and then injected), in vivo labeled cells (i.e., the cells were labeled while the inside the body of the animal), and cells expressing the GFP. The animals have been either a fully-furred adult mouse or rat. The data that has been acquired by each flow cytometer is presented and discussed below. In addition, recommendations are made for future work, including physical modifications to the cytometers, performance-measuring tests, software modifications, and future experiments.

Single-slit, Single-color In Vivo Flow Cytometer

The single-slit, single-color in vivo flow cytometer system has been used to acquire data involving white blood cells labeled with Cy-Chrome, and red blood cells, leukemia cells, and prostate cancer cells labeled with DiD. The first application of the single-slit, single-color system was to determine the differences in flow charcteristics between venular flow and arterial flow for human red blood cells circulating within a mouse[9]. Specifically, it was desired to determine how velocity was distributed among the human red blood cell population (i.e., number of cells versus velocity) within each type of blood vessel. To this end, circulating human red blood cells were isolated, labeled, and injected into a mouse. The labeling process was done ex vivo rather than in vivo to avoid labeling other types of cells, as well as to better control the number of labeled human red blood cells within the bloodstream of the animal. It was desired to inject enough to yield statistically valid data, but not so many that significant overlap of the signals from the cells would occur. The human red blood cells were labeled with 0.1-mM DiD. (DiD is a lipophilic dye that binds to cell membranes.) The mouse was a young adult female Balb/C mouse. The labeled human red blood cells were injected in

200 µl saline into the tail vein. As stated in Chapter 2, the vasculature from which data was acquired was in the ear of the animal. Figure 4.1 depicts typical data traces obtained. Figure 4.1A is a representative voltage trace for labeled human red blood cells flowing through an artery, while Figure 4.1B is for flow through a vein. The traces were displayed after implementing a 50-point moving window averaging on the original data.

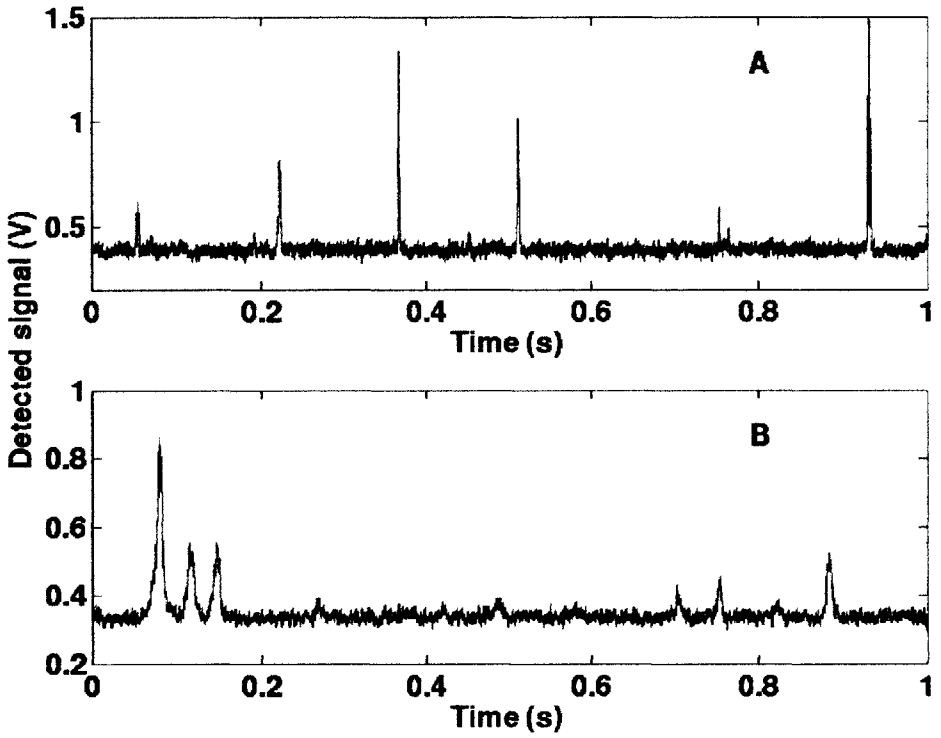

Figure 4.1 Representative smoothed traces of fluorescently labeled human red blood cells flowing through an artery (A) and a vein (B) in the ear of a mouse.

Figure 4.2 shows the pulse-width distribution for labeled cells detected in the two types of blood vessels. The histograms represent the number of peaks per minute detected with a specific full-width-half-maximum. The black is for arterial flow and the gray is for venous flow. Velocity (or an approximation of it) was obtained by dividing the cell diameter, which was 8 microns, by the fwhm values. Thus, for arterial flow, approximately 1.5% had a velocity of 8 mm/sec, 89.5% were in the range 2.00 mm/sec to 4.00 mm/sec, and 10.0% were at or below 1.60 mm/sec. For venous flow, approximately

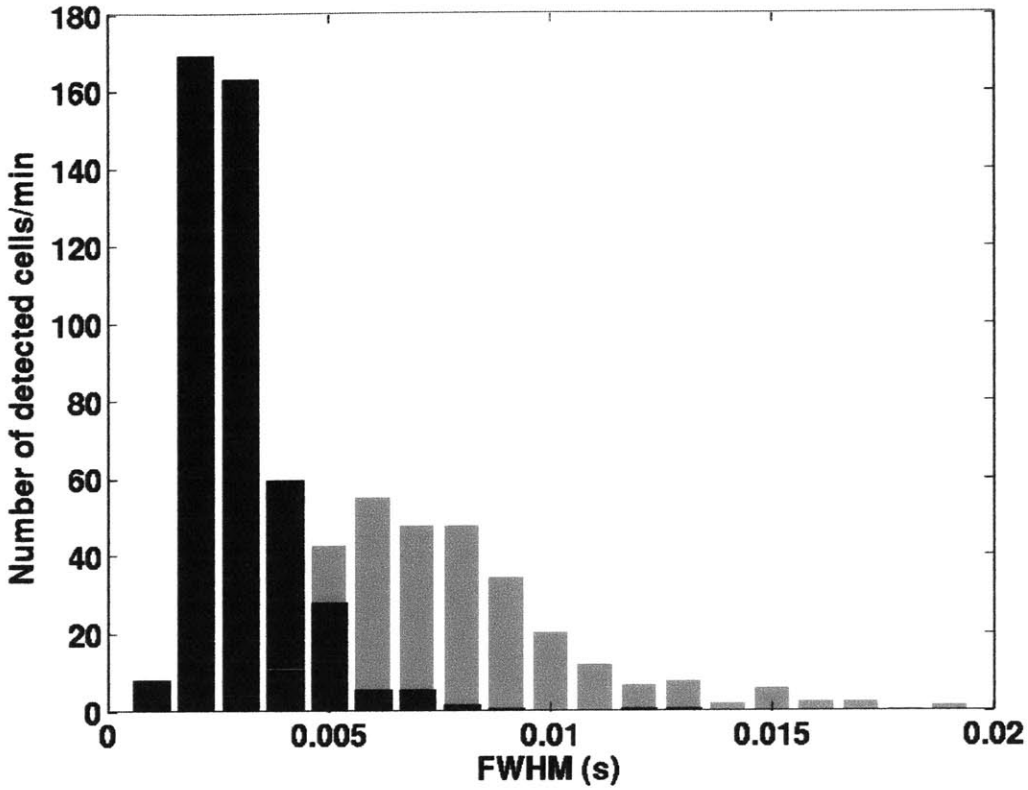

Figure 4.2 Histograms representing the number of peaks with a specific FWHM representing DiD-labeled red blood cells per minute in an artery (black) and a vein (gray) of a mouse ear

17.5% had a velocity between 1.60 mm/sec and 2.67 mm/sec, 67.5% were between 0.8mm and 1.60 mm/sec, and 15% were below 0.8 mm/sec. These velocities measured were consistent with previous reports using optical Doppler tomography[10]. Note, however, unlike the in vivo flow cytometer, optical Doppler tomography cannot provide information on the percentage of cells within a particular velocity range.

Building upon this velocity distribution study, it was next desired to contrast the kinetics of the circulating ex vivo labeled red blood cells with the kinetics of mouse white blood cells labeled in vivo with fluorescently tagged antibody[9]. To this end, 20 μg of rat antimouse CD45 monoclonal antibody labeled with Cy-chrome was injected through the tail vein of another young adult female Balb/C mouse. Because the white blood cells express the CD45 antigen on their surface, the antibody labeled the circulating white blood cells. In vivo flow cytometry measurements were performed on both mice, one

with the labeled red blood cells, and the other with the labeled white blood cells. For the labeled red blood cells, the number detected remained constant over a three-day recording period. In contrast, the number of white blood cells varied in a dynamic and rapid way. Figure 4.3A summarizes the labeled red blood cell count, and Figure 4.3B summarizes the white blood cell count. Error bars are shown as well. The results for the

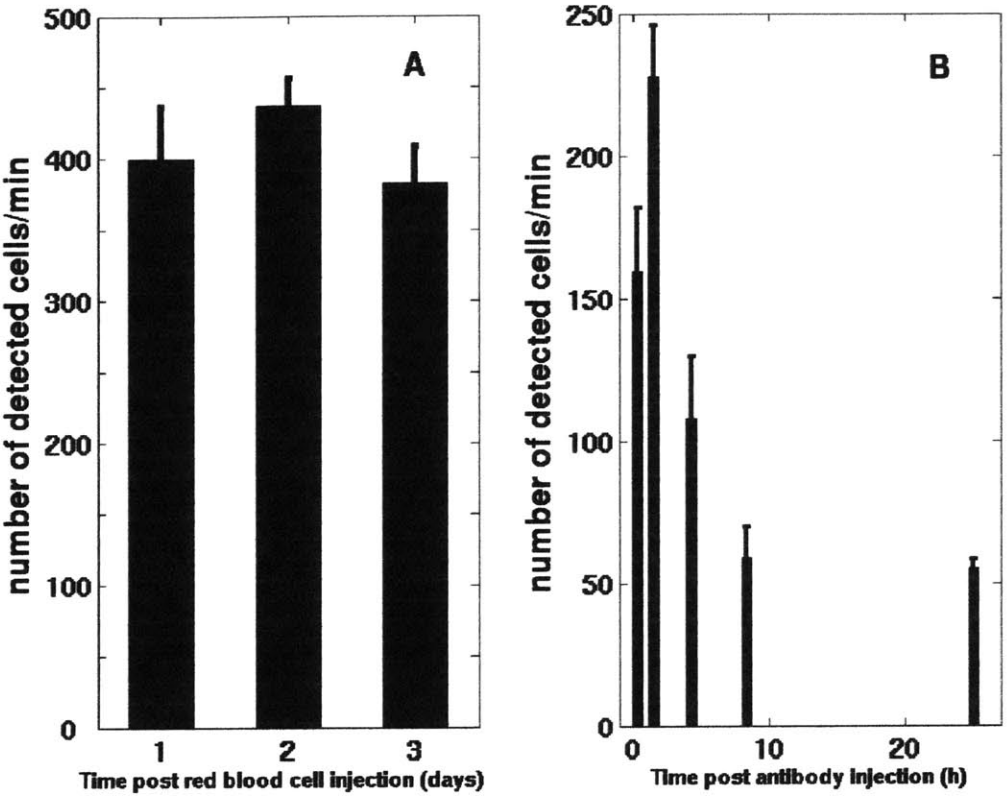

Figure 4.3: (A) The number of human red blood cells, labeled ex vivo and injected in the mouse circulation through the tail vein, flowing through a mouse ear artery remains constant for a period of three days. (B) In contrast, the number of white blood cells labeled in vivo with a fluorescently-tagged antibody, vary in a dynamic and rapid way from the time of antibody injection.

red blood cells indicate that the labeling process employed to label the red blood cells did not cause their depletion. However, for the white blood cells, it appears that the antibody binding labeling process resulted in approximately 75% depletion of the labeled white blood cells within the first 8 hours. Similar results were obtained by labeling the mouse CD4+ T-cell population with a fluorescent anti-CD4 antibody (see Figure 4.4 below).

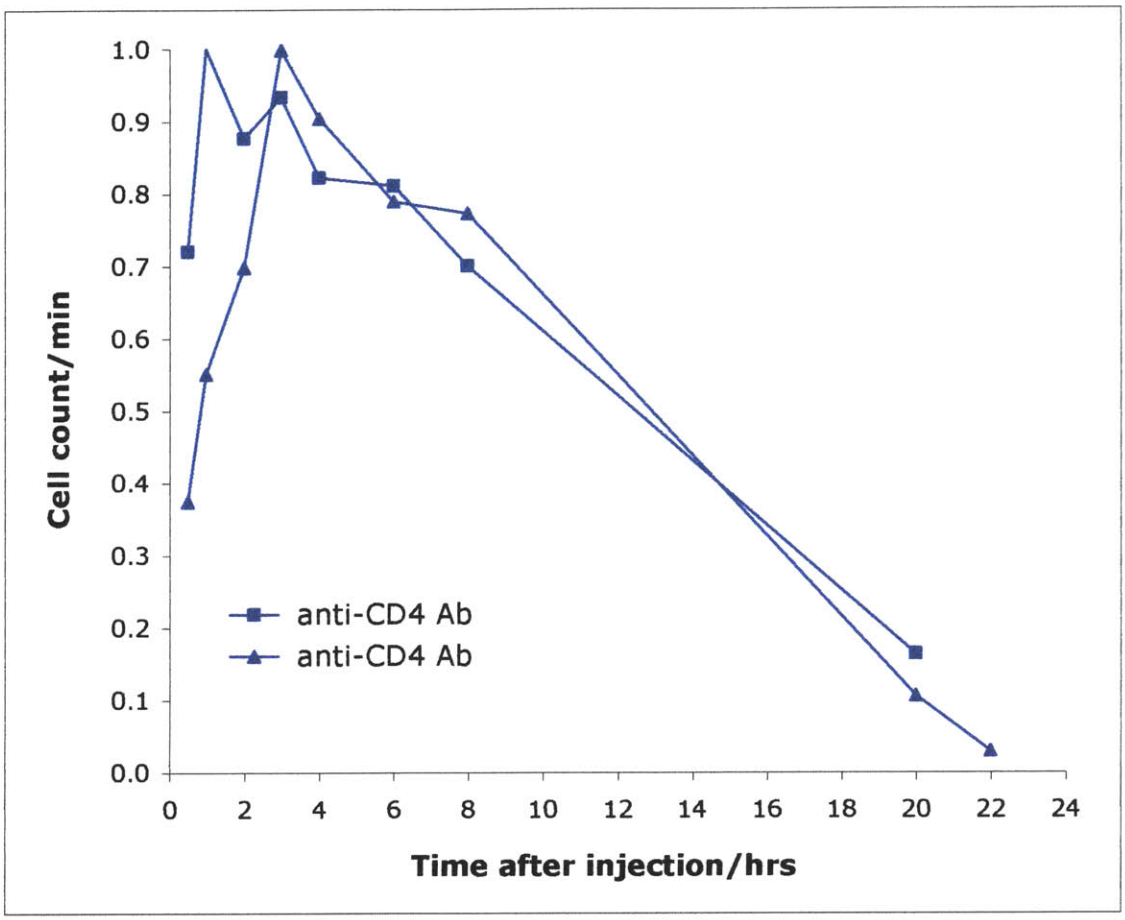

Figure 4.4 Temporal population profile of mouse CD4+ T-cells labeled with a fluorescent anti-CD4 antibody

For possible future experiments involving white blood cells, this depletion of white blood cells due to antibody binding is unacceptable. Following the kinetics of white blood cell depletion, especially in the first several hours, is important in the study of tissue and organ transplantation and autoimmune diseases such as rheumatoid arthritis and AIDS, as well as response to therapeutic manipulation such as antibody therapy. Consequently, efforts are now being directed to find a new method for in vivo labeling of white blood cells that does not result in significant depletion of the leukocytes. For example, since it is commonly believed that depletion of the leukocytes labeled with either the conjugated CD4 or CD45 antibody is primarily due to Fc-mediated mechanisms associated with the antibodies, approaches are being pursued to either mask or remove the Fc region of the antibodies. Thus, the in vivo flow cytometer has proven to be an effective tool, first in bringing to light the presence of flaws associated with the white blood cell labeling

processes originally used, and second in quickly and efficiently testing different cell-labeling methods.

The in vivo flow cytometer has also been used to study the depletion kinetics of prostate cancer cells[3]. The prostate cancer cells studied were human LNCaP cells and rat MLL cells. SCID mice and Copenhagen rats were used in the study. The depletion kinetics of the different prostate cancer cell lines was examined for dependence on cell line and type of animal host.

The SCID mice were male and 4 to 6 weeks old. Their body weight was approximately 25 grams. The Copenhagen rats were male as well, 6 to 8 weeks old, and weighing 50 to 75 grams. The prostate cancer cells were injected through the tail vein. Approximately one million labeled cancer cells per 20 grams of body weight were injected. Count measurements using the single-slit, single-color in vivo flow cytometer were taken five to fifteen minutes after injection, as well as 1, 2, 4, and 8 or 10 hours later, and 1, 2, 3, and 5 days later.

The figure below, Figure 4.5, is a representative data trace, where each spike is

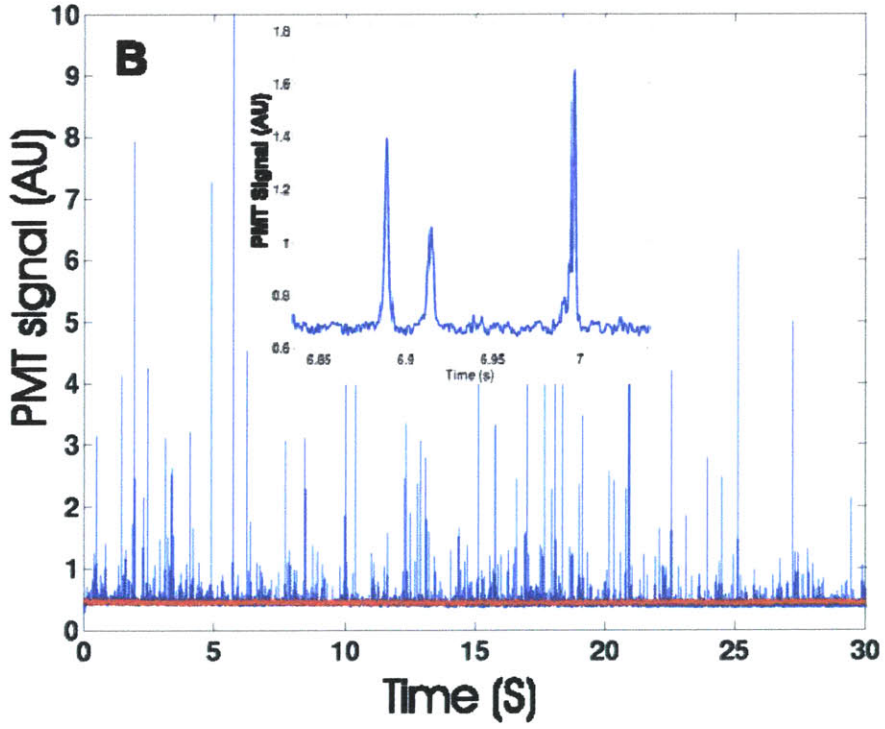

Figure 4.5 Representative signal trace. Control trace is shown in red. Inset shows fluorescence peaks in more detail.

created by the fluorescence burst resulting from a DiD-labeled prostate cancer cell traversing the 632 nm excitation slit. The variation in the intensity of the recorded peaks is probably due to the corresponding variations in the intensity of fluorescence staining of individual cells. The control trace is shown in red. As mentioned in the previous chapter, the control trace is obtained by acquiring data at the data acquisition site before injection of the labeled cancer cells. It gives a measure of the background noise level at the data acquisition site, and is used to help discern cell signal from noise. Figure 4.6 is a scatter plot of the height and width (fwhm) of the fluorescence peaks from the labeled cancer cells, as well as the noise peaks of the control data. The points due to fluorescence peaks are shown in blue, those due to noise are shown in red. The green

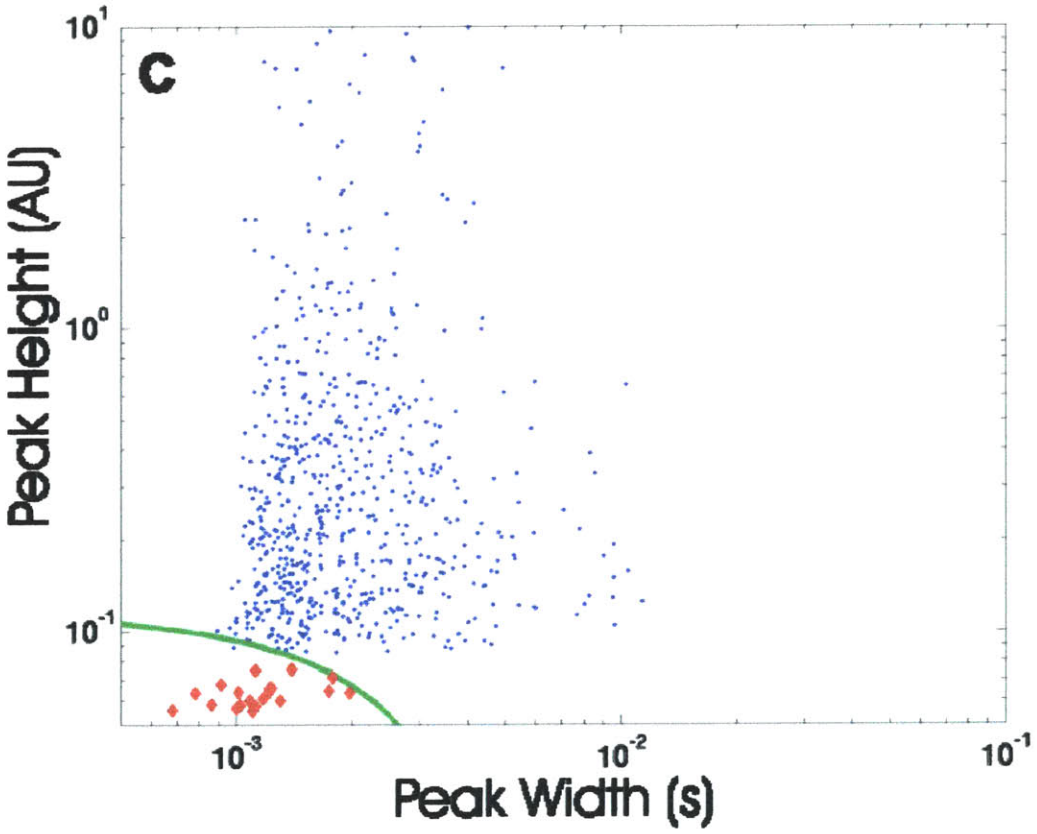

Figure 4.6 Scatter plot of the fluorescence peaks and noise peaks. The green line is the user-specified boundary between noise and signal.

line is drawn by the user and is obtained when the control data alone is displayed on the scatter plot. It represents the user's judgment of the boundary on the scatter plot between

noise and signal. This line is the line that is defined by the slope and intercept parameters discussed in the previous chapter. (Note that the above scattered plot is a log-log plot, resulting in a curved line rather than straight line.)

The number of cells detected per minute versus time after injection for both types of cancer cells in both hosts is shown below in Figure 4.7. All measurements are

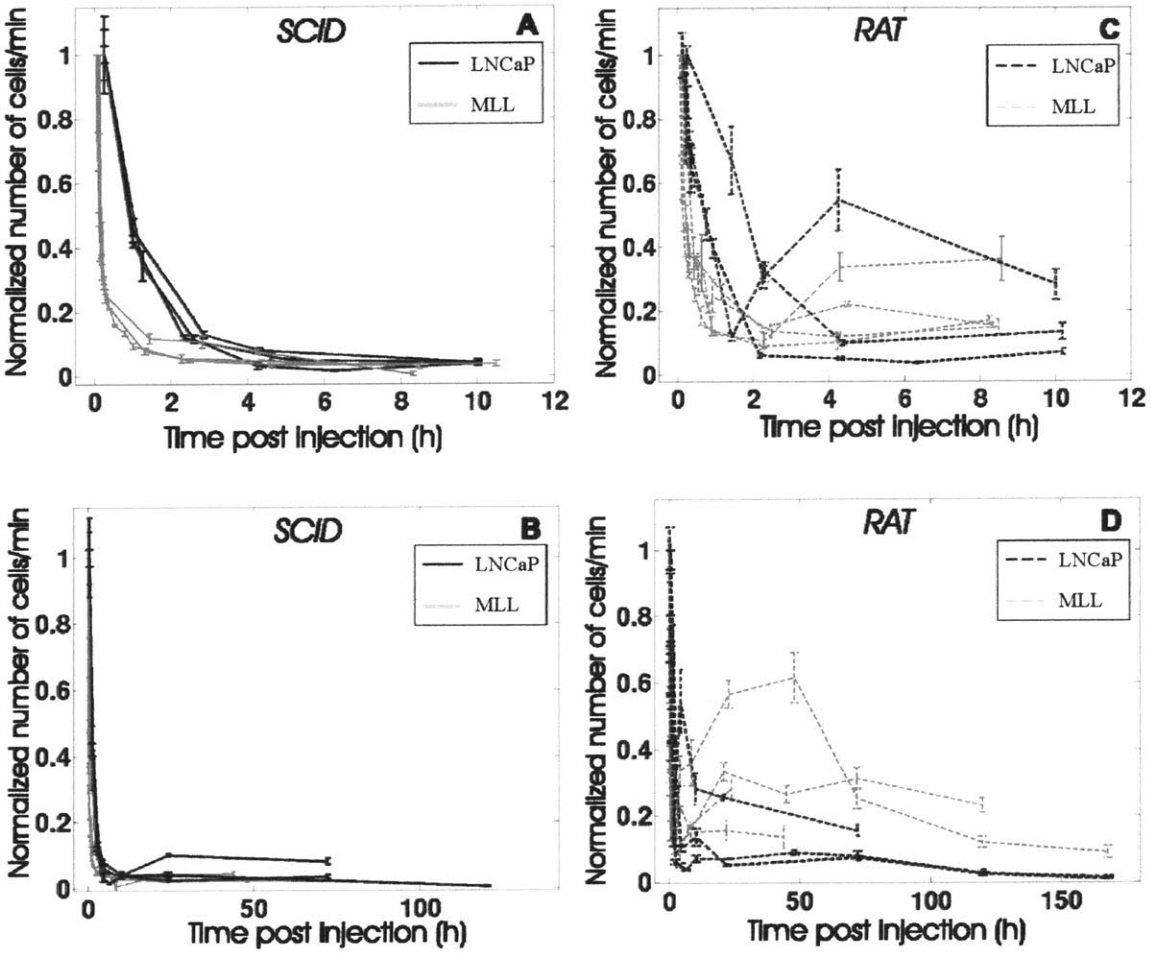

Figure 4.7 Depletion kinetics of LNCaP (black lines) and MLL (gray lines) prostate cancer cells in SCID mice (panels A and B) and Copenhagen rats (panels C and D).

normalized to the number of cancer cells detected immediately after injection, which was the time point that yielded maximum number of detected cells. The black lines show the depletion kinetics of the LNCaP prostate cancer cells, and the gray lines show the depletion kinetics of the MLL prostate cancer cells. Panels A and B are the results obtained with the SCID mouse as the host, and panels C and D are the results obtained

with the Copenhagen rat as the host. As one can see by inspection of the plots, more than 80% of both types of prostate cancer cells are depleted from both types of host within the first 2 to 4 hours after injection of the cancer cells. In addition, the acquired data indicates that the more metastatic MLL cells deplete from both hosts at a statistically significant faster rate than the LNCaP cells. It is also seen that the depletion of both types of cancer cells within the SCID mice follows a systematic and reproducible pattern, whereas for the Copenhagen rats as the host, there is greater variability in the depletion pattern. For the SCID mice, once the initial depletion of both types of cancer cells takes place, the number of circulating cancer cells remains consistently low for at least several days. For the Copenhagen rats, after the initial depletion, there is a reappearance of both types of cancer cells most of the time, but to varying degree levels. In addition, in some cases, this reappearance trend eventually disappears followed by depletion again, whereas in other cases the reappearance trend continues.

Information on the distribution of fluorescence peak heights and widths (fwhm) for each type of cancer cell and for each host was also obtained, allowing for extraction of further information. Histograms of the normalized distribution for LNCaP cells of the fluorescence peak heights and the widths (fwhm) are shown below in Figure 4.8. From the peak height histogram, one can see that the peak height distribution of the labeled LNCaP cancer cells did not change with time for the Copenhagen rat as the host animal, but did for the SCID mouse as the host animal. This indicates that there was no significant changes in the level of cell labeling for the rats. In the case of the SCID mice, it is noticed that there are fewer high intensity peaks observed at the 24-hour time point. It is possible that these high intensity peaks correspond to cell aggregation, which become arrested in organs such as the lungs. The fwhm histogram indicates that the velocity of the LNCaP cells, estimated by dividing the cell diameter of 10 microns by the fwhm, ranged primarily between 1.0 mm/sec to 10.0 mm/sec for the data acquisition site chosen. Similar results were obtained for the MLL cells.

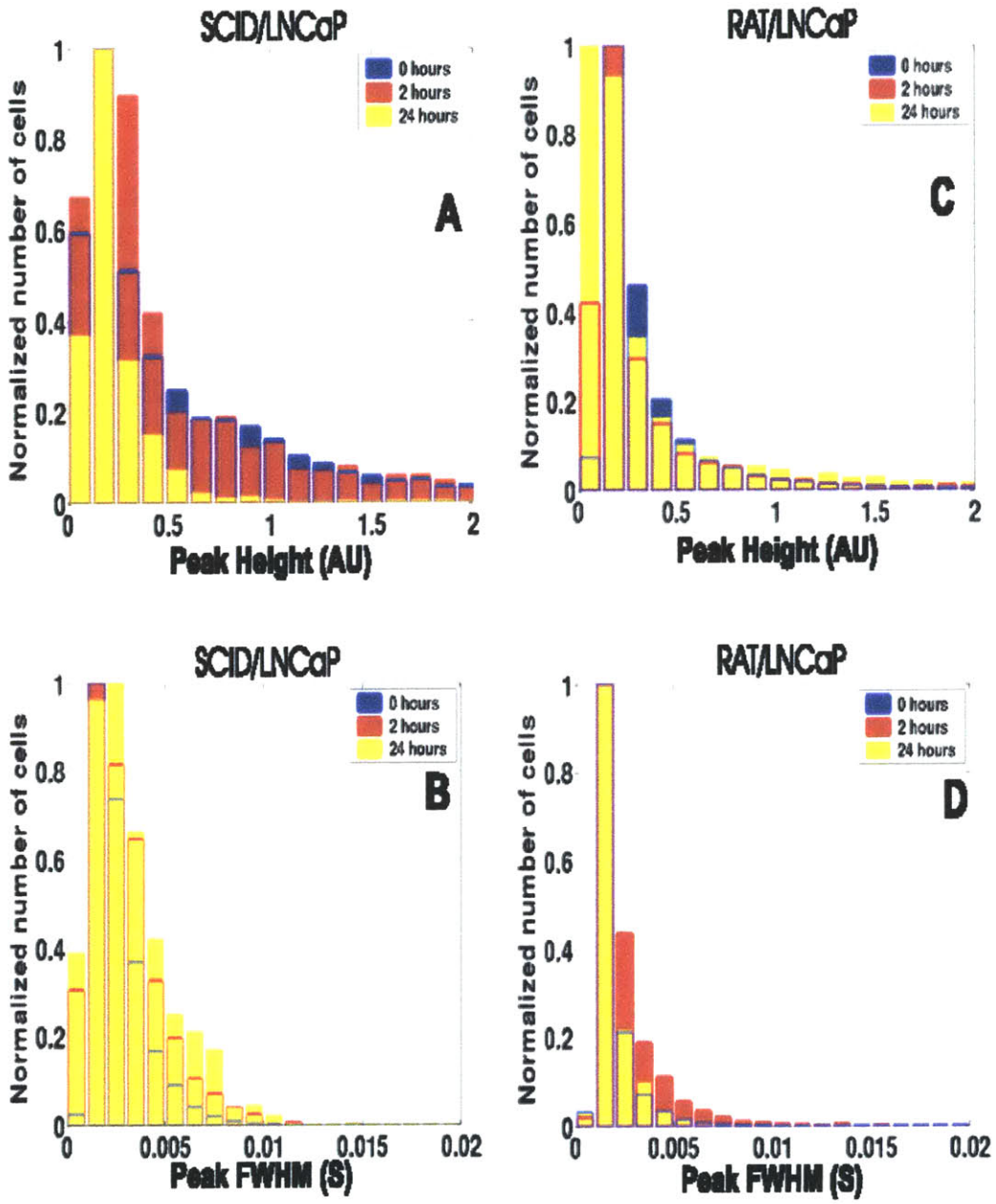

Figure 4.8 Histograms of peak height and fwhm for LNCaP cells for both hosts.

Work was also performed to obtain an estimate of the correlation between number of labeled cells present in the body of the animal and number of cells detected per minute[11]. This was important to verify that the in vivo flow cytometer had sufficient sensitivity to provide valid data on population trends. To this end, labeled LNCaP cancer cells were injected into a SCID mouse in quantities of 10^3, 10^4, 10^5, and 10^6, and then in vivo flow cytometer measurements performed immediately after injection. The in vivo

flow cytometer measurements were performed immediately after injection because the disparity between the number of cells injected and number of cells in circulation would be at its minimum at this point in time. The results are shown in Figure 4.9. The plot

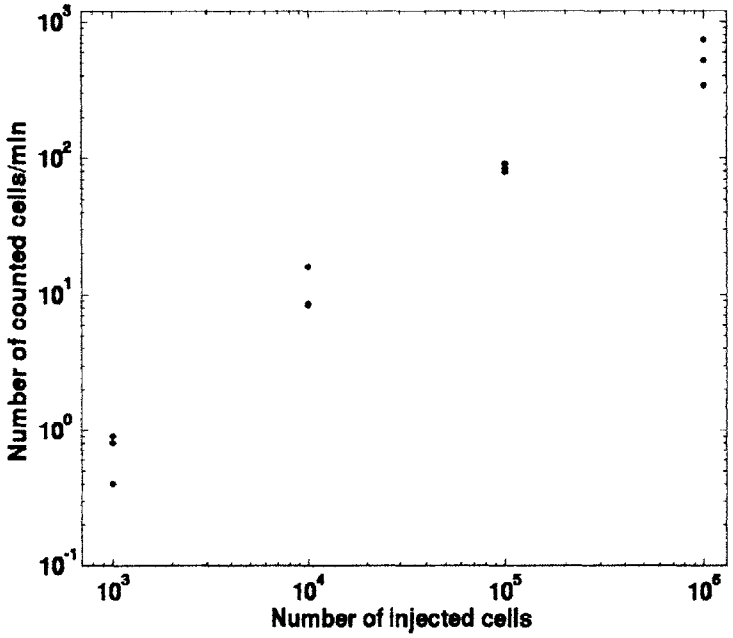

Figure 4.9 Detected cells per minute versus number of injected cells

clearly shows a consistent relationship between number of injected labeled cells and number of cells/min detected, indicating that the in vivo flow cytometer has sufficient sensitivity to provide valid temporal population profiles over a wide range of labeled cell populations in the body of the host animal. Minimal extrapolation of the data indicates that labeled cell populations below 1000 can be detected.

Two-slit, Two-color In Vivo Flow Cytometer

The two-slit, two-color system, capable of single-slit and two-slit detection, as well as single-color and two-color detection, has been utilized in a variety of ways. One of the first applications was detection of cells expressing the EGFP gene using the 473 nm channel. This experiment was chosen to be performed first to investigate the difficulties associated with acquiring data from the 473 nm channel, and to develop a data acquisition site selection algorithm which would result in reliable cell detection at the

shorter wavelengths. The shorter wavelengths tend to be absorbed and scattered to a greater extent by the skin and hair than the longer wavelengths, and the amount of resulting autofluorescence tends to be greater as well. The autofluorescence spectrums obtained for the 473 nm channel and the 632 nm channel, acquired using the setup shown in Figure 4.10 when focused on the skin and artery of a mouse, prove this point.

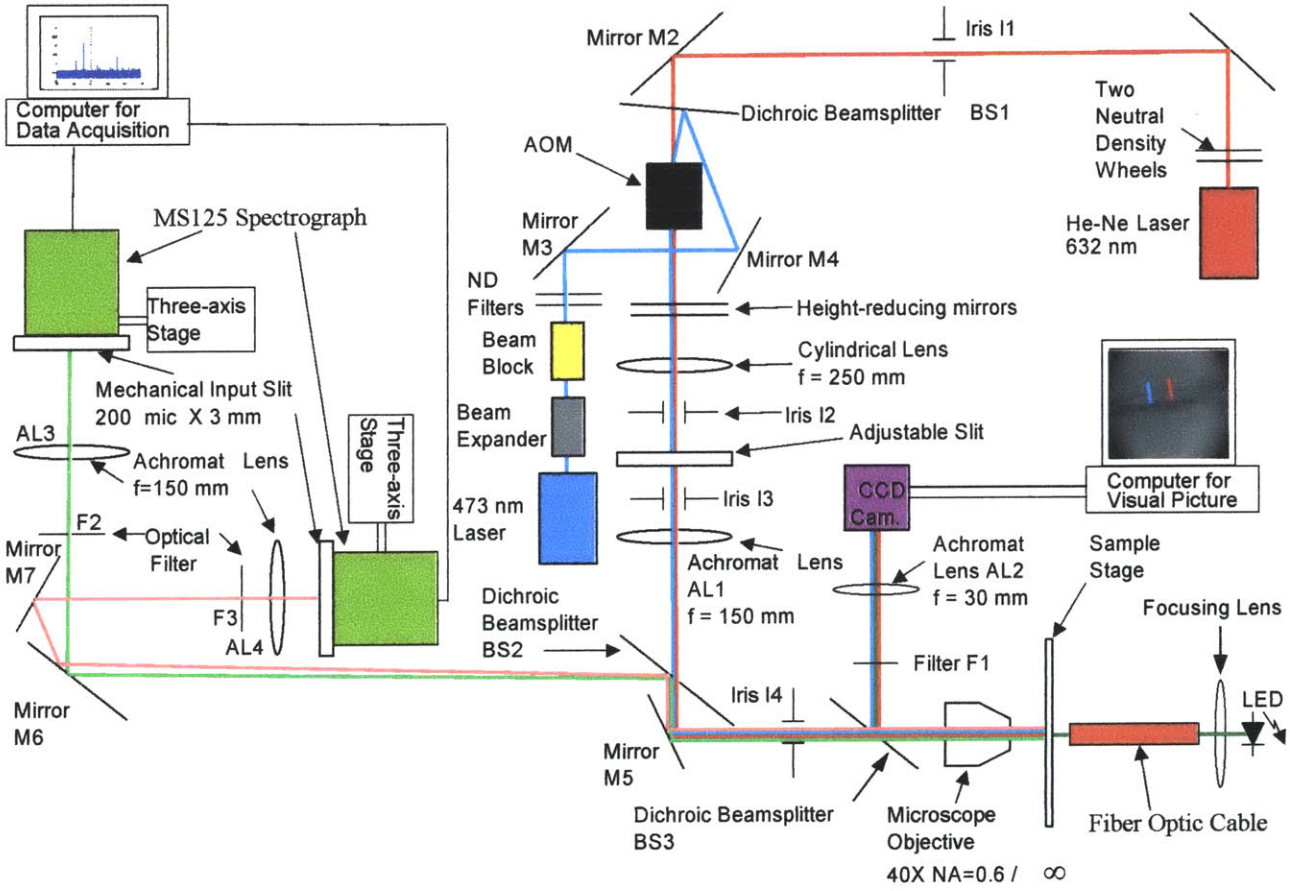

Figure 4.10 Setup to acquire autofluorescence spectrums. MS125 Spectrographs replace photomultiplier tubes and mechanical slits replaced with mechanical input slits of spectrographs.

Figure 4.11, the autofluorescence spectrum of the skin using the 473 nm laser, is much larger in intensity than that shown in Figure 4.12, the autofluorescence spectrum obtained using the Helium-neon laser.

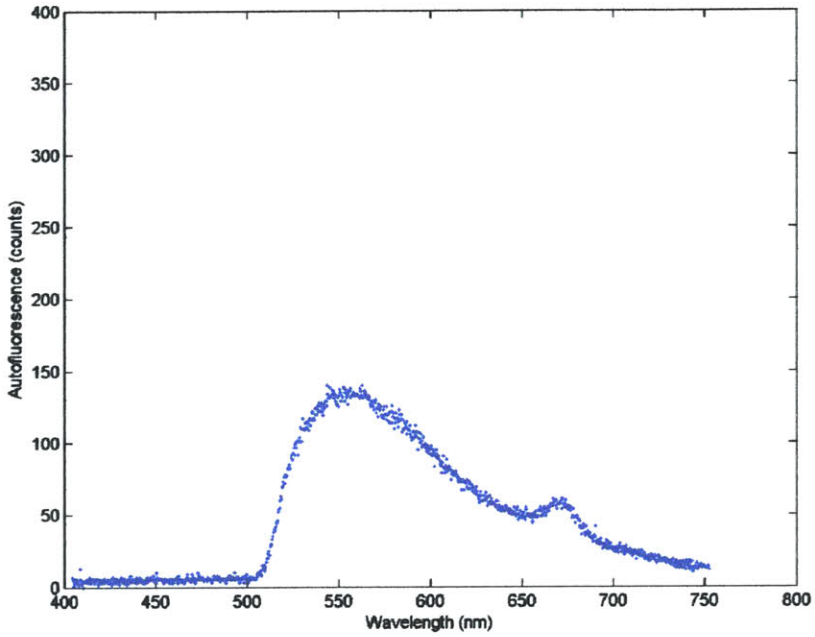

Figure 4.11 The resulting autofluorescence due to the 473 nm laser focused on the skin of a SCID mouse. Long pass filter that turns on at 510 nm is detector filter

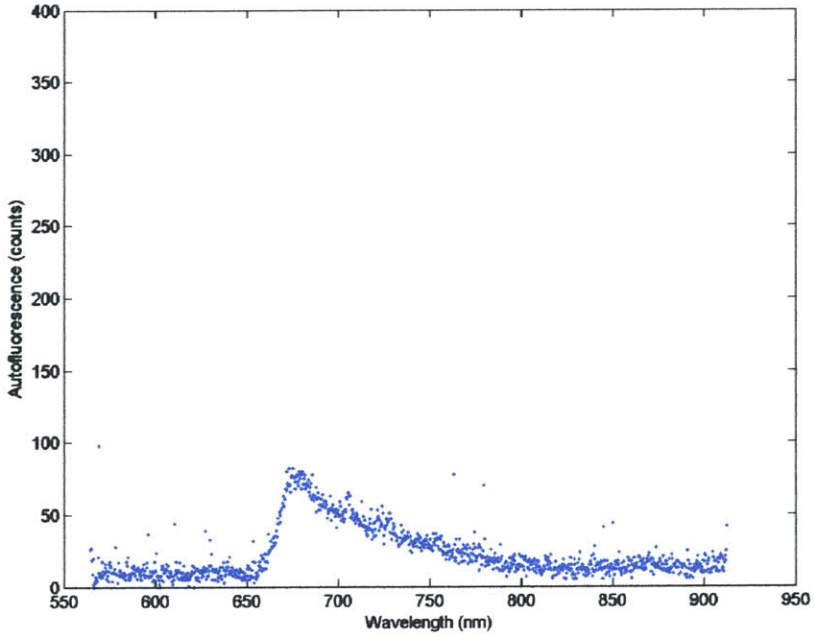

Figure 4.12 The resulting autofluorescence due to the 632 nm laser focused on the skin of a SCID mouse. Long pass filter that turns on at 660 nm is detector filter

The same is true when focused on an artery. Figure 4.13, the autofluorescence spectrum obtained with the 473 nm laser focused on an artery, is significantly greater in intensity than the autofluorescence spectrum resulting from using the 632 nm laser, shown in Figure 4.14.

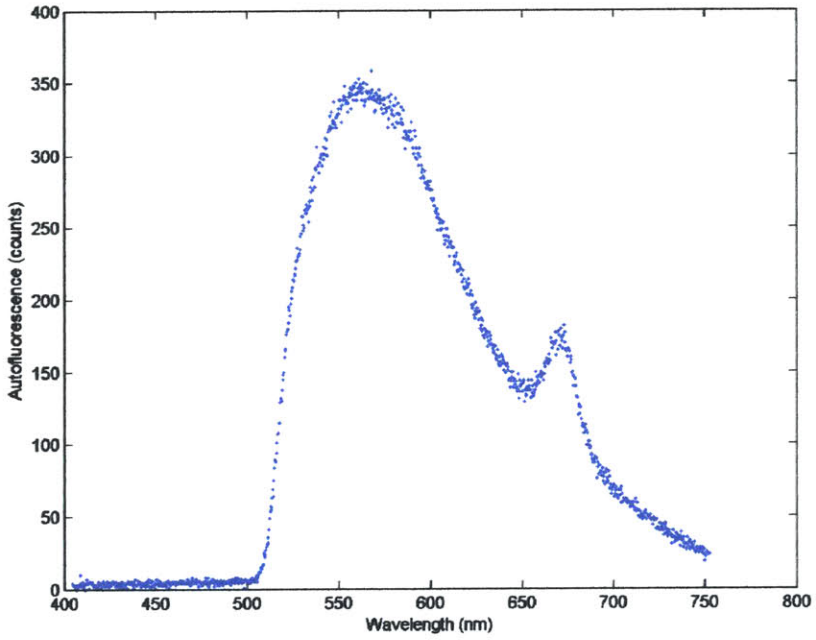

Figure 4.13 The resulting autofluorescence due to the 473 nm laser focused on an artery of a SCID mouse. Long pass filter that turns on at 510 nm is detector filter

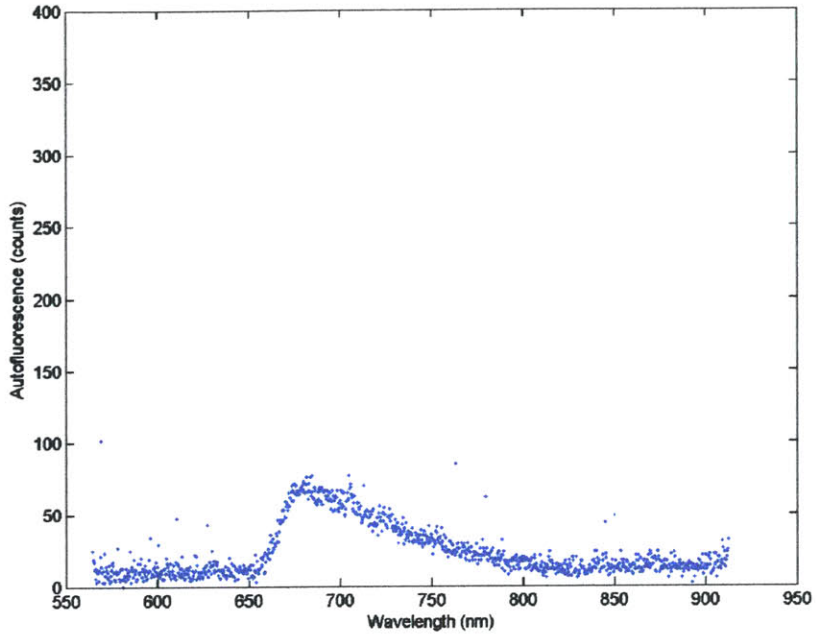

Figure 4.14 The resulting autofluorescence due to the 632 nm laser focused on an artery of a SCID mouse. Long pass filter that turns on at 660 nm is detector filter

Mice containing circulating cells expressing the EGFP gene were selected to avoid having to perform injections of labeled cells or injection of labeling attachments, the effects of which only last for several hours before re-injection is required. The EGFP expressing cells were $MHCII^+$ cells. MHCII is a molecule in several subpopulations of leukocytes. The EGFP gene is coupled to the MHC molecule. The level of expression of the EGFP gene (i.e., the level of fluorescence of the cell) is dependent on MHCII expression. The more the MHCII molecule is expressed, the greater the level of fluorescence of the cell. The absorption and emission spectrum created by the EGFP gene are shown below (from the Clontech website http://www.bdbiosciences.com/clontech/archive/OCT99UPD/RFP.shtml), along with other fluorescent proteins. Peak excitation occurs at 489 nm, peak emission at 508 nm. The mice containing these EGFP- expressing leukocytes were those of the C57LB6

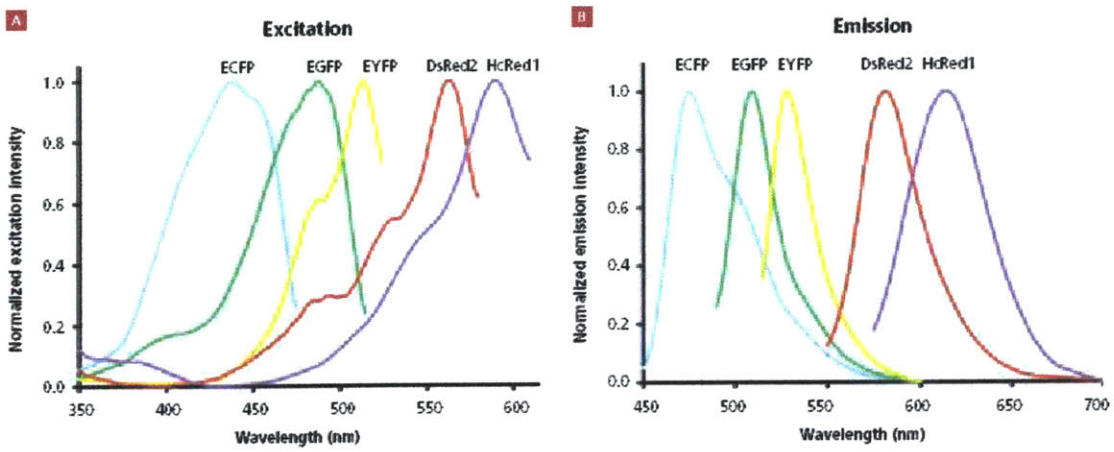

Figure 4.15 Excitation (A) and emission (B) spectra of various fluorescent proteins

strain. After much experimentation (Dr. Mehran Poureshagh and John Novak), it was discovered that successful cell detection took place if one avoided, in location of the 473 nm excitation slit, hair, hair follicles, sebaceous glands, melanin patches, and capillary flow. In addition, it was found that photobleaching the data acquisition site by the 473 nm laser for several minutes before data acquisition helped reduce autofluorescence background. Application of the above resulted in traces from which useful information, such as cell count, cell velocity, and population distribution information could be extracted. A representative trace is shown below.

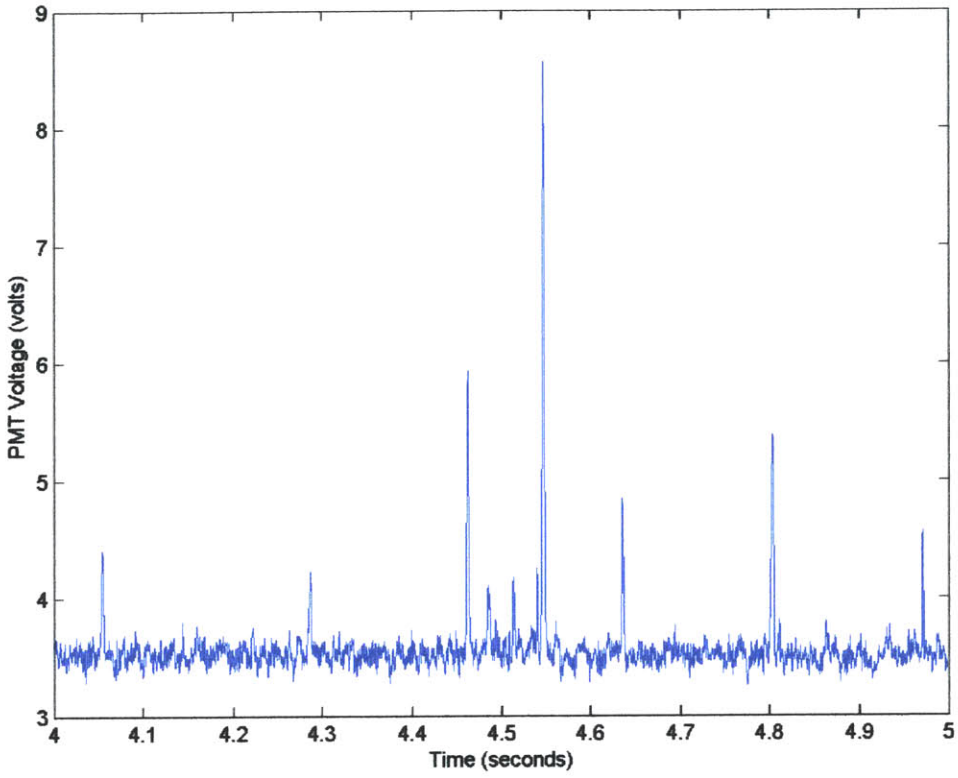

Figure 4.16 Detection of MHCIIGFP^{+} cells circulating in the blood stream of an C57LB6 strain mouse

With the data acquisition site selection criterion for the 473 nm channel worked out, attention was now focused on the 632 nm channel. Although it was known, based upon experience with the single-slit, single-color system, that acquisition of data from the 632 nm channel was significantly less sensitive to the physical aspects of the skin than the 473 nm channel, nevertheless, the site selection criterion developed for the blue channel applied to the 632 nm channel was investigated. To this end, human red blood labeled with DiD were injected into a BALB/c mouse, and data acquired from the 632 nm channel. The results, as expected, were excellent. The background noise level was small and the signal strength was strong. A representative trace is shown below.

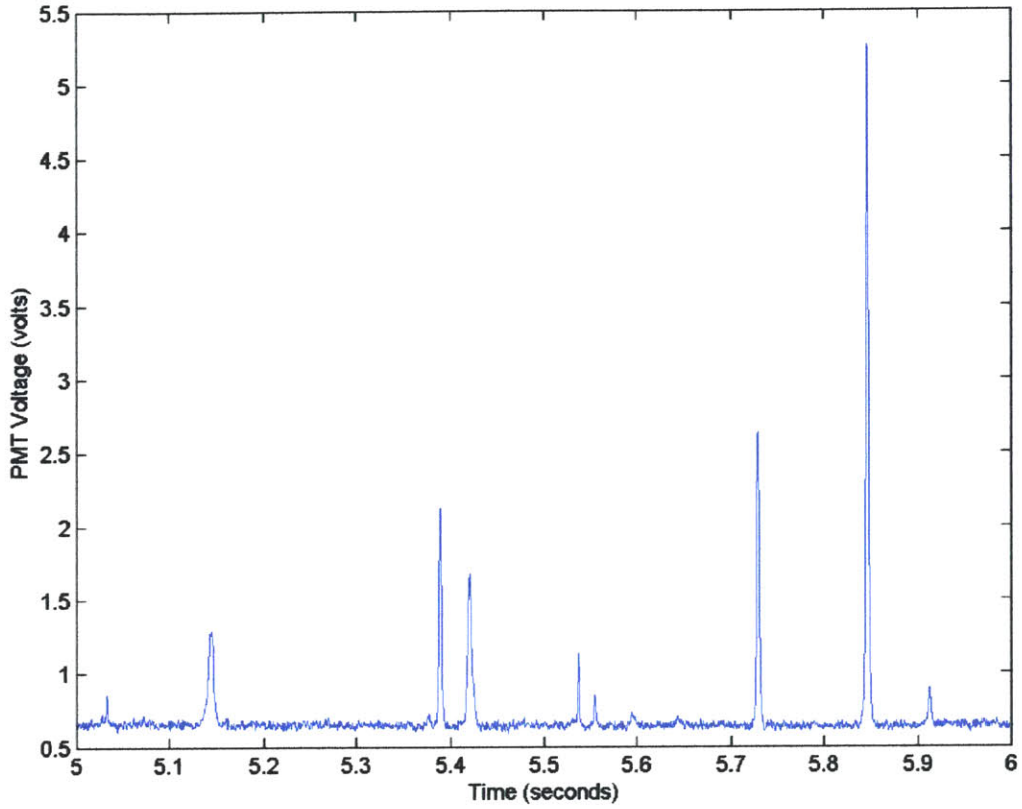

Figure 4.17 Detection of human red blood cells labeled with DiD circulating in the blood stream of a BALB/c mouse.

Thus, having developed a site selection criterion applicable to both channels, as well as verified the proper independent operation of both channels, attention was now focused on investigating the usage of the system in the two-slit, two-color mode. For this mode of operation, it remained to be verified that the device could properly acquire data from both channels simultaneously, and that the detection capability of both channels was sufficient for various fluorescence sources. To this end, the following experiments were performed in the two-slit, two-color mode:

1) Detection of T-cells of a BALB/c mouse co-labeled with DiD and Cell Tracker (both fluorescent labels from Molecular Probes).

2) Detection of red blood cells co-labeled with DiD and DiO (both fluorescent labels from Molecular Probes) in a BALB/c mouse.

3) Detection of leukemic cells co-labeled with Calcein and DiD (both fluorescent labels from Molecular Probes) in a BALB/c mouse.

4) Detection of leukemic cells co-labeled with DiD and Cell Tracker in a SCID mouse.

5) Detection of MHCIIGFP$^+$ cells, some also labeled with Cychrome (from Molecular Probes) conjugated to antiCD45 antibody in a C57LB6 strain mouse.

DiD and Cychrome are fluorescent labels for the 632 nm (red) channel. Cell Tracker, DiO, Calcein, and EGFP provide fluorescence in the 473 nm (blue) channel. The perfect results would be a one-to-one correspondence in cell detection between the two channels for experiments 1,2,3, and 4, and cell detection in the blue channel for cell detection in the red channel for experiment 5, but not the converse. For experiment 5, as mentioned above, a subpopulation of the MHCIIGFP$^+$ cells were labeled with cychrome. The experimental results obtained for the five different experiments performed are shown below.

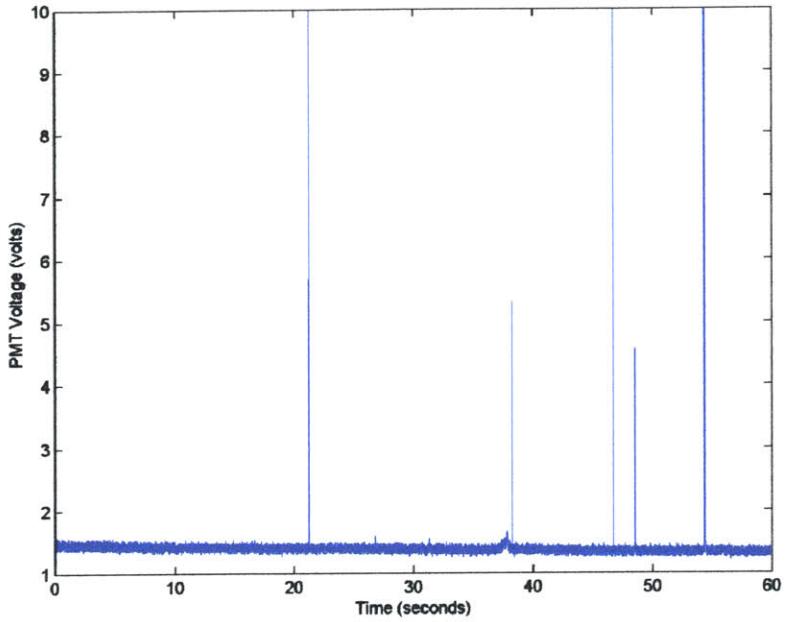

Figure 4.18 Detection by the 473 nm (blue) channel of T-cells of a BALB/c mouse co-labeled with DiD and Cell Tracker . Cell tracker is the fluorescence source for the blue channel.

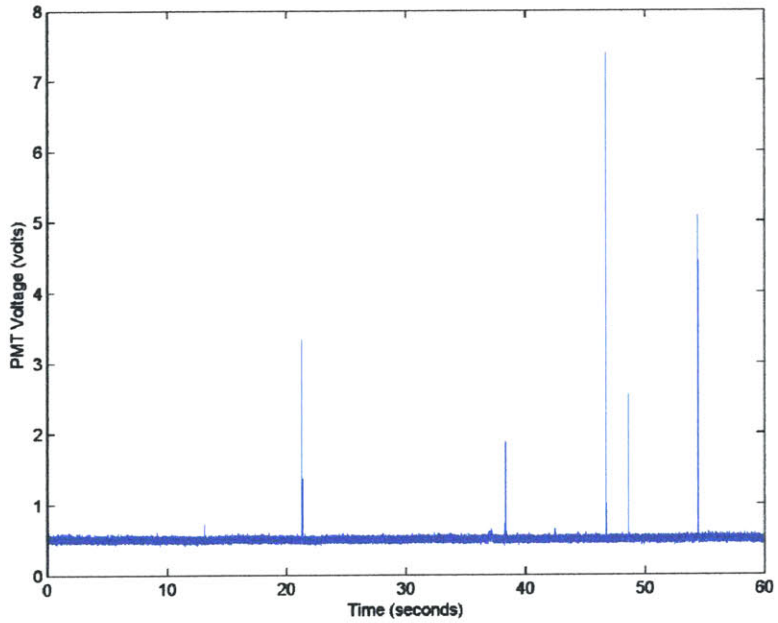

Figure 4.19 Detection by the 632 nm (red) channel of T-cells of a BALB/c mouse co-labeled with DiD and Cell Tracker . DiD is the fluorescence source for the red channel.

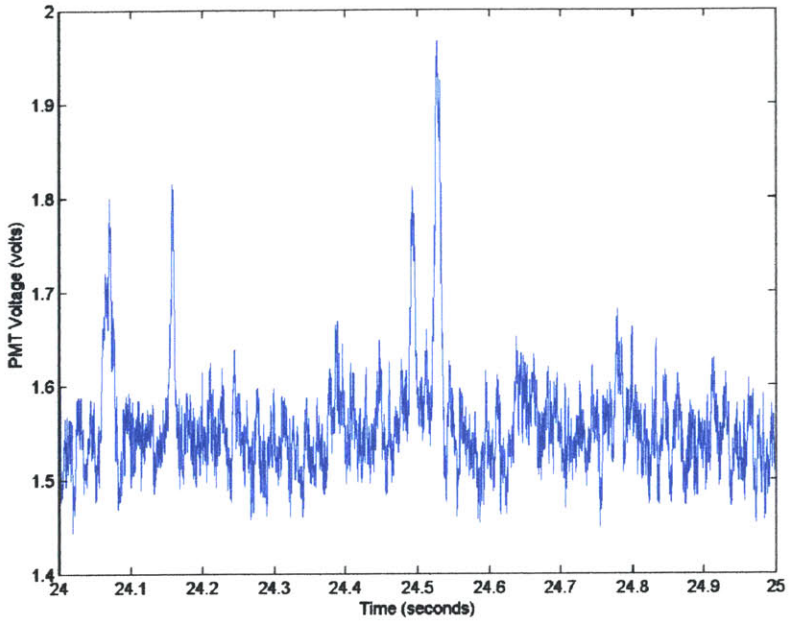

Figure 4.20 Detection by the 473 nm (blue) channel of red blood cells of a BALB/c mouse co-labeled with DiD and DiO . DiO is the fluorescence source for the blue channel.

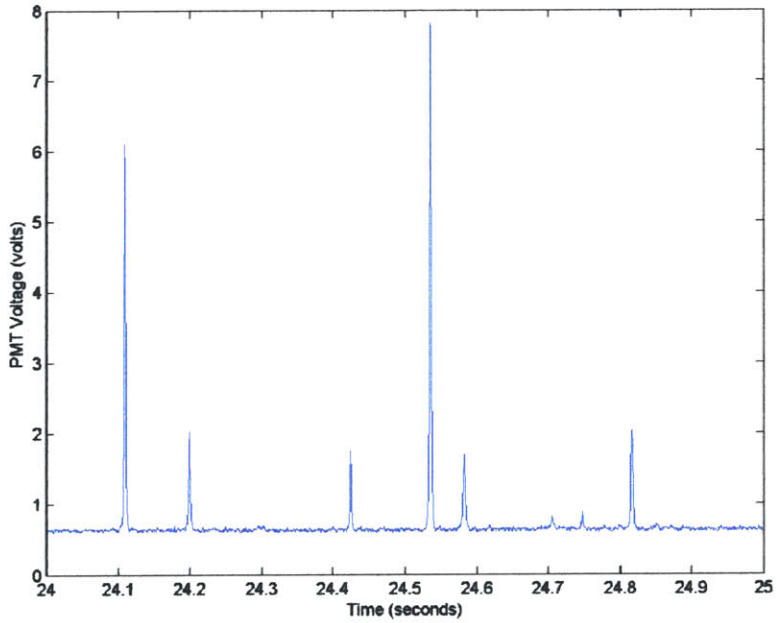

Figure 4.21 Detection by the 632 nm (red) channel of red blood cells of a BALB/c mouse co-labeled with DiD and DiO . DiD is the fluorescence source for the red channel.

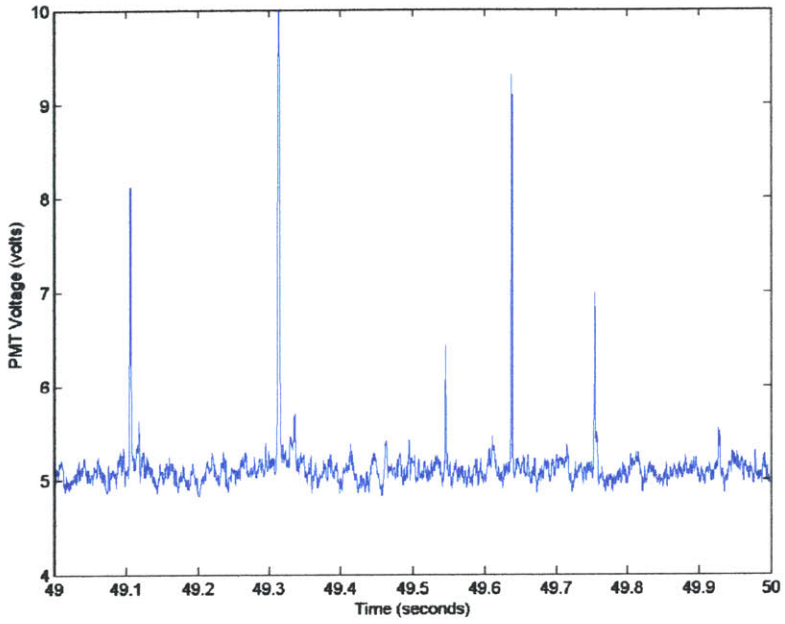

Figure 4.22 Detection by the 473 nm (blue) channel of leukemia cells co-labeled with Calcein and DiD in a BALB/c mouse. Calcein is the fluorescence source for the blue channel.

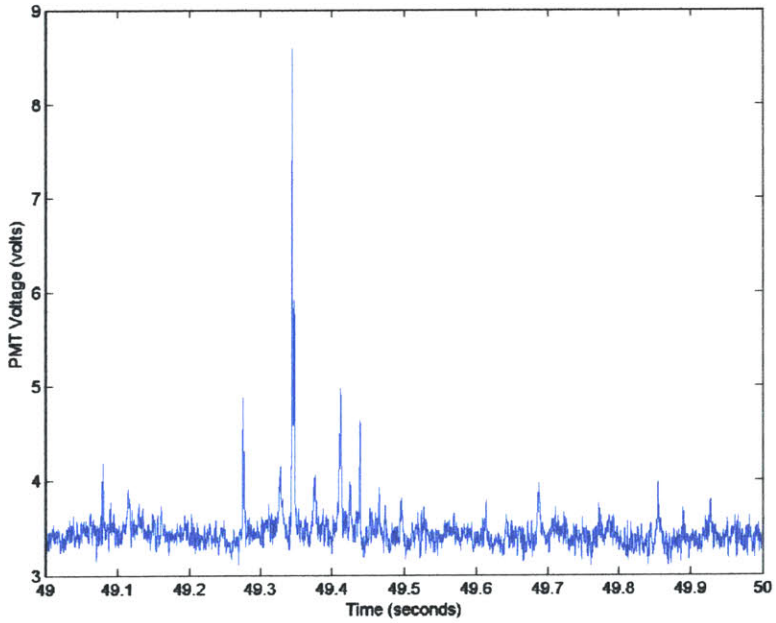

Figure 4.23 Detection by the 632 nm (red) channel of leukemia cells co-labeled with Calcein and DiD in a BALB/c mouse. DiD is the fluorescence source for the red channel.

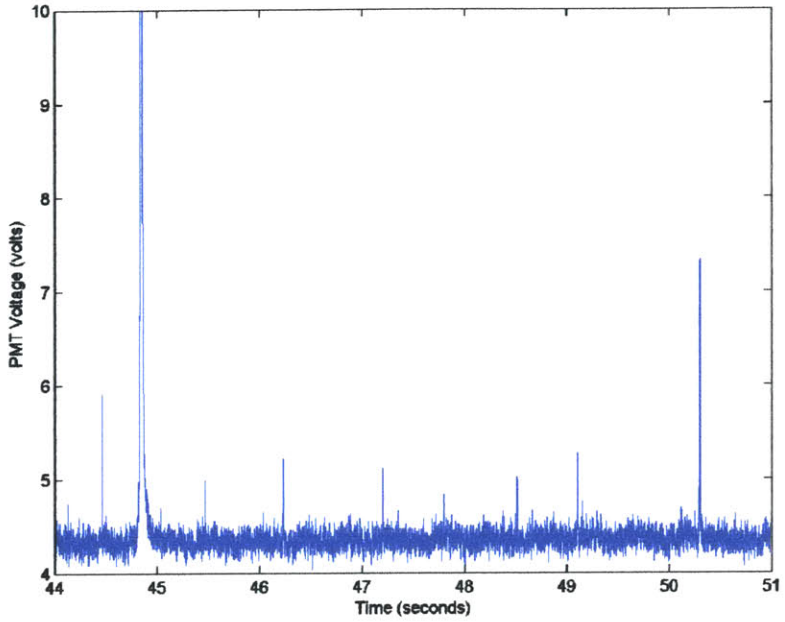

Figure 4.24 Detection by the 473 nm (blue) channel of leukemia cells co-labeled with DiD and Cell Tracker in a SCID mouse. Cell tracker is the fluorescence source for the blue channel.

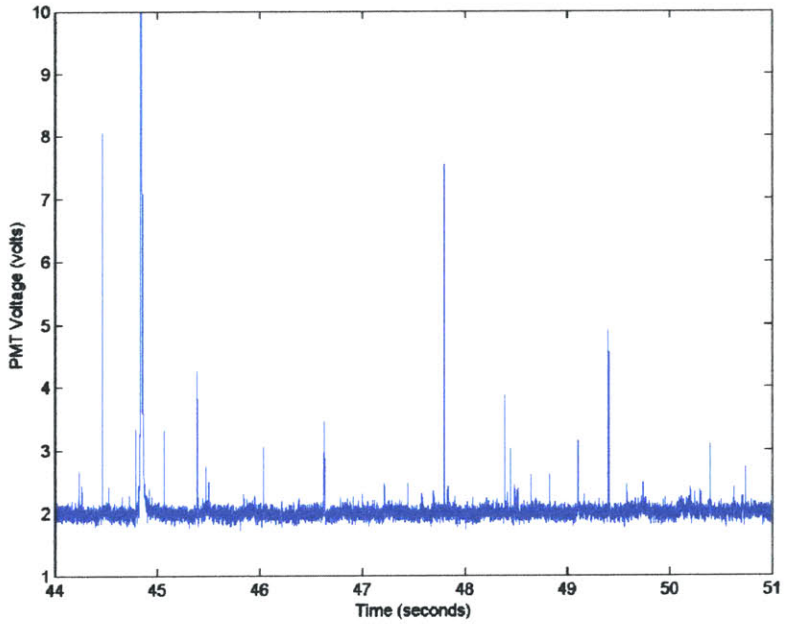

Figure 4.25 Detection by the 632 nm (red) channel of leukemia cells co-labeled with DiD and Cell Tracker in a SCID mouse. DiD is the fluorescence source for the red channel.

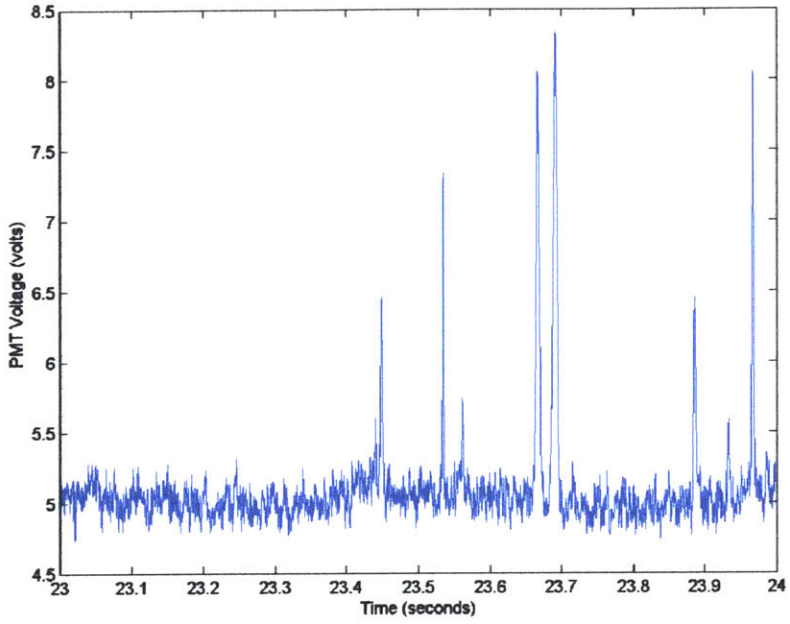

Figure 4.26 Detection by the 473 nm (blue) channel of MHCIIGFP$^+$ cells, some also labeled with Cychrome, in a C57LB6 strain mouse. The EGFP is the fluorescence source for the blue channel.

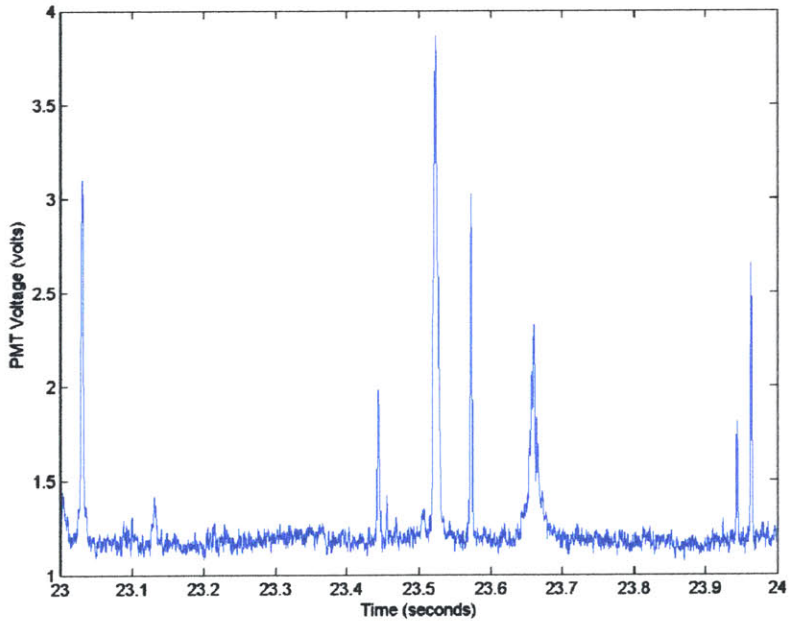

Figure 4.27 Detection by the 632 nm (red) channel of MHCIIGFP$^+$ cells labeled with Cychrome, in a C57LB6 strain mouse. Cychrome is the fluorescence source for the red channel.

The results of the first experiment, detection of T-cells co-labeled with DiD and Cell Tracker, were almost perfect. The five peaks in the blue channel (detection using Cell Tracker) corresponded perfectly to the peaks in the red channel (detection using DiD). The only discrepancy was that the peak in the red channel at 13 seconds did not have a corresponding peak in the blue channel. For the second experiment, where red blood cells were co-labeled with DiD and DiO, there was little correspondence between the channels, and many more fluorescence due to DiD than DiO were detected. The third experiment involving the co-labeling of leukemia cells with calcein and DiD showed little detection correlation between the channels as well. However contrary to the previous experiment, many cells were detected in each channel over the 60-second data acquisition period. For the fourth experiment, where leukemia cells were co-labeled with cell tracker and DiD, there was moderate correlation between the two channels, with a moderate number of cells detected in both channels. For the last experiment, where a subpopulation of $MHCIIGFP^{+}$ cells were labeled with cychrome (conjugated to antiCD45 antibody), many cells were detected in both channels, with some temporal stretches in which the expected results were obtained (where a detection in the blue channel corresponded to a detection in the red channel but not the converse).

Thus, based on the results of the first experiment, and partially on the results of the fourth and fifth experiment, it appears that the device is capable of acquiring data from both channels simultaneously. However, the above experiments also seem to indicate that the results of the labeling process vary depending on the fluorescence sources being used, as well as the cells being labeled. Also, detection in the 473 nm (blue) channel, although generally sufficient, appears to be less sensitive than detection in the 632 nm (red) channel. Concerning this last observation, this is probably due to the skin and blood absorbing and scattering both the excitation signal and the fluorescence signal to a much greater extent in the blue channel than in the red channel. In addition, as already shown, the autofluorescence from the skin is more intense at the shorter wavelengths, creating more background noise. Further work will have to be performed to improve the reliability of the labeling process, and further work can be performed to improve the signal-to-noise ratio of the blue channel. Solution of this problem to obtain

reliable and consistent detection simultaneously in both channels is highly desirable, since it will then allow for many new biological experiments to be performed.

The last aspect of the cytometer investigated was performance in the two-slit, single-color mode. For this experiment, the Helium-neon laser and the AOM were activated to create two excitation slits of wavelength 632 nm. In addition, the mechanical slit in front of the PMT of the 632 nm channel was replaced with a variable-width slit to allow two streams of fluorescence photons, one from each excitation slit, to reach the PMT. This type of setup was used instead of two mechanical slits and two detectors because it provided a more severe test for the capability of the system, since the variable-width slit was only partially confocal with the two excitation slits at the sample stage. Human red blood cells labeled with DiD and injected into a BALB/c mouse were the source of signal. The results obtained are shown below. As one can see by inspection of

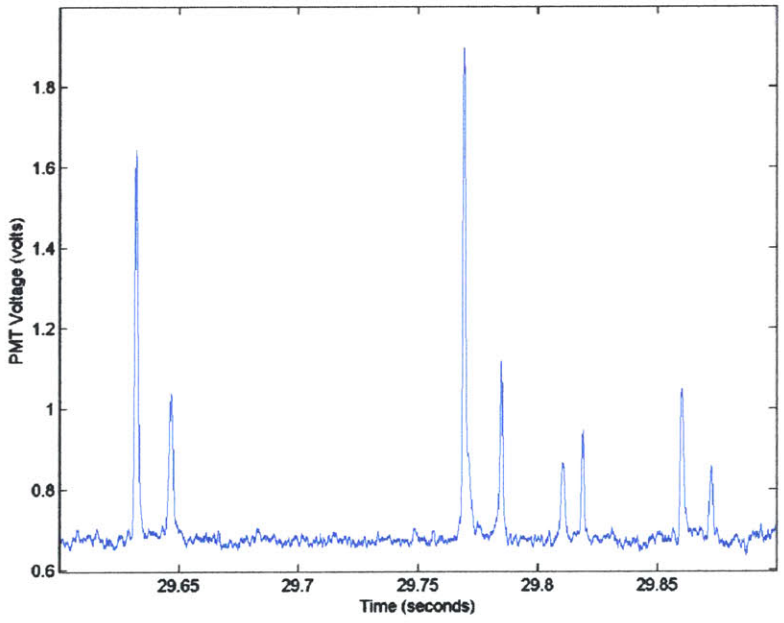

Figure 4.28 Detection of human red blood cells labeled with DiD by the cytometer using two excitation slits at 632 nm wavelength in a BALB/c mouse.

the plot, the cytometer functioned as desired in this mode of operation, providing a voltage signal when a labeled cell passed through an excitation slit. Note that the time required for the labeled cells to travel between the excitation slits can be read from the plot, allowing for, if one knows the separation distance between slits, the calculation of

the cell velocity. For this experiment, the separation distance was approximately 50 microns, yielding a cell velocity of 2.5 mm/sec. Note that operation of the cytometer in this two-slit, single-color mode will increase in value as more fluorescence sources which can be excited by the same wavelength of light and have non-overlapping emission spectra become available. For example, DiD and DiR, lipophilic dyes that can both be excited by the Helium-neon laser, overlap strongly from 650 nm to 700 nm, as shown below. Figure 4.29 is the absorption/emission spectrum of DiD, and Figure 4.30 is the emission spectrum of DiR for wavelengths greater than 660 nm (obtained from leukemia cells on a slide labeled with DiR using the MS125 spectrograph). This overlap precludes usage of the two dyes simultaneously, since signal from the DiR will be recorded in the DiD channel.

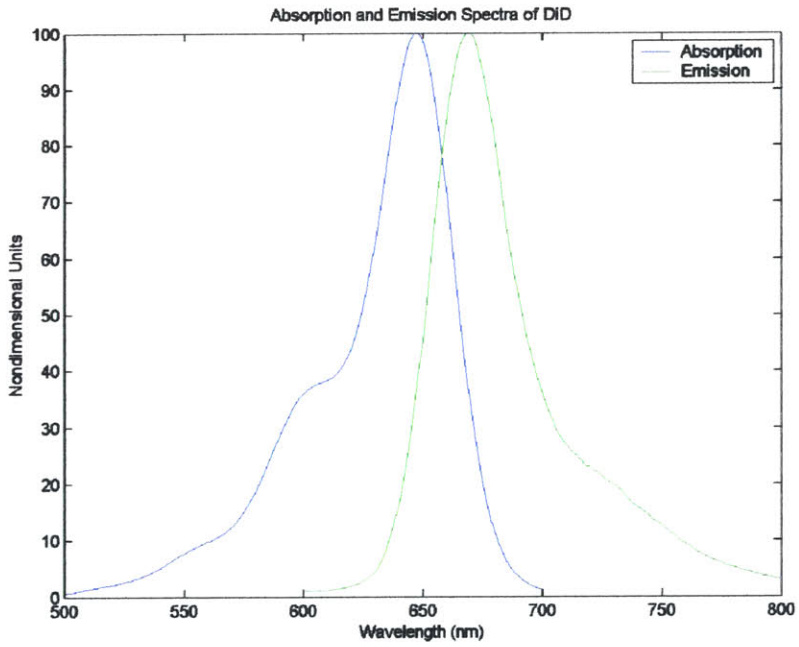

Figure 4.29 Absorption and emission spectrum of the lipophilic dye DiD.

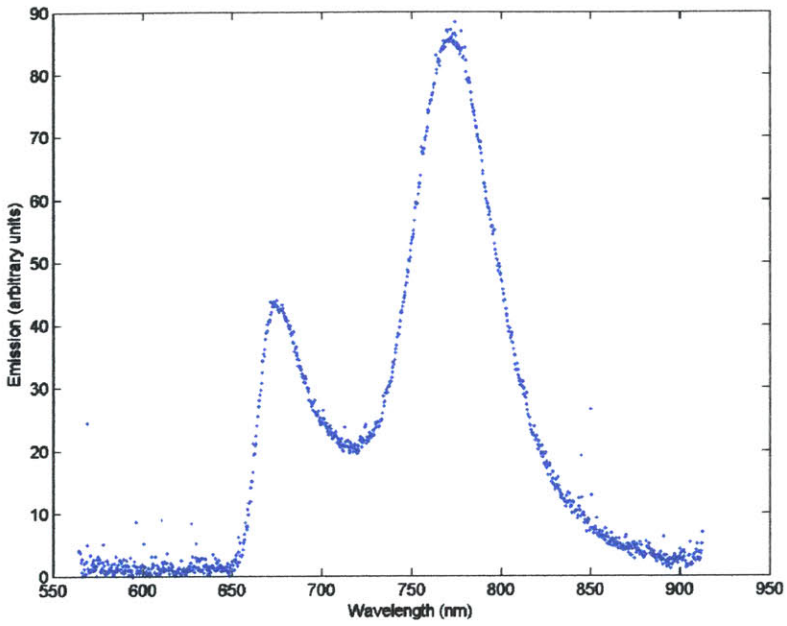

Figure 4.30 The emission spectrum of the lipophilic dye DiR for wavelengths greater than 660 nm (obtained from leukemia cells on a slide labeled with DiR using the MS125 spectrograph).

Summary and Recommendations for Future Work

In summary, two devices have been developed which combine the concepts of standard flow cytometry and confocal detection to allow for acquisition of flow cytometric data in vivo without the need to extract a blood sample. The first device that was developed, termed the single-slit, single-color in vivo flow cytometer, can detect and count cells that possess a fluorescent label which can be excited by a Helium-neon laser. In addition to being able to quantify the number of fluorescently labeled circulating cells as a function of time, other information such as cell velocity and population distribution of cell velocity, cell signal height, and cell signal width can be extracted. The host animals used thus far from which data have been acquired include SCID and BALB/c mice, as well as Copenhagen rats. The animals were full-grown, fully-furred adults and did not require any type of special preparation such as shaving of the fur. The data acquisition site was the ear of the animal. The cells that have been labeled and studied thus far include mouse red blood cells and leukocytes, and human red blood cells, leukemia cells, and prostate cancer cells. Several biological studies such as contrasting

the kinetics of circulating ex vivo labeled red blood cells with mouse white blood cells labeled in vivo with a fluorescently tagged antibody, investigation of the correlation of metastatic potential with circulating cell count for leukemia cells and prostate cancer cells, and development of a labeling method for leukocytes that does not result in significant depletion of the labeled leukocytes have been carried out.

The second device that was developed is termed the two-slit, two-color in vivo flow cytometer. It is a more advanced version of the single-slit, single-color cytometer in that it can be operated in several different modes. The device can provide one excitation slit comprised of laser light at either 473 nm or 632 nm, it can provide two excitation slits comprised of laser light at either 473 nm or 632 nm, or it can provide two excitation slits with one at wavelength 473 nm and the other at 632 nm. In the two-slit mode, the fluorescence photons created be each excitation slit can be channeled into separate detectors or into the same detector. At present, cell detection sensitivity involving usage of the 632 nm laser is greater than that involving usage of the 473 nm laser. This is probably due to greater scattering and absorption by the skin and blood at the shorter wavelengths, as well as increased noise level due to autofluorescence of the skin. Further research can be performed to find ways to improve the signal-to-noise ratio for the blue channel. One option is to shift to a slightly longer excitation wavelength, which can be accomplished by replacing the present laser with an argon laser, which emits at 488 nm. This will allow for more excitation energy to reach the fluorescence sources (and, hence, less autofluorescence by the skin) and more absorption by the fluorescence sources. Although the improvement will probably be only minimal given the similarity of the optics of the skin and the absorption characteristics of the fluorescence sources at 473 nm and 488 nm, it will still result in a better signal-to-noise ratio. Another option is to alter the physical dimensions of the mechanical slit confocal with the excitation slit. The present mechanical slit is 3 mm long, which is what is required for an excitation slit 100 microns long, the originally planned slit length. However, since it has been found that a 30 micron blood vessel diameter is usually best of which to acquire data, and the length of the excitation slit is made to match the blood vessel diameter (via a variable mechanical slit at the focal length of the cylindrical lens), the mechanical slit can be reduced in length by a factor of about three. Although this will reduce some signal, this

disadvantage will (probably) be negligible since most of the signal enters at +/- 0.5 mm about the longitudinal center, and virtually all light entering beyond these dimensions is noise. The slit width, which controls the depth of focus (i.e., detection), can also be investigated for optimum performance. Like the mechanical slit length, it might be possible to further reduce the slit width to further reduce noise without any serious compromise in signal strength. A third option is to change the microscope objective to a higher numerical aperture objective, and to reduce further the amount of under-filling of the aperture of the objective by the excitation laser light. The higher numerical aperture objective will result in greater photon gathering capability by the system, and a further reduction in the effective numerical aperture at the laser light input side of the objective will result in an excitation slit width narrower and more constant in dimension across the blood vessel. The slight penalty that will have to be paid for a narrower slit width is that the present cylindrical lens will have to be replaced with one of a longer focal length. This will be necessary to demagnify further the horizontal dimension (i.e., the dimension parallel to the optical table) of the laser beam that ultimately becomes the excitation slit width. A fourth option that can be investigated is conversion of the excitation section from single-photon to two-photon. Two-photon excitation, based on the principle that two photons of longer wavelength light simultaneously absorbed by a fluorochrome molecule cause fluorescence to occur that would normally require a single photon at a much shorter wavelength, allows for the elimination of the mechanical slit in front of the detector and, hence, for collection of all signal photons. This should result in an increase in the signal-to-noise ratio, since the level of autofluorescence reaching the detector should remain approximately unchanged. (For single-photon excitation, the autofluorescence originating outside the excitation volume confocal with the mechanical slit in front of the detector is blocked by the mechanical slit. For two-photon excitation, autofluorescence will only occur within the excitation volume, where the density of excitation photons is sufficiently high.) In addition, this method of excitation will allow for deeper penetration into tissue, since, in general, to wavelengths of approximately 1100 nm, longer wavelength light penetrates tissue more deeply than shorter wavelength light.

In addition to possible physical changes to the system, improving cell labeling is also a viable option. New dyes and new fluorescent sources such as quantum dots, which have higher quantum yields and extinction coefficients than existing dyes, are becoming commercially available. The higher quantum yields and extinction coefficients result in more fluorescence photons being created for a given excitation intensity. In addition, labeling techniques to increase the number of fluorescent sources attached to the cells can be investigated. For example, tyramide signal amplification (TSA)— sometimes called CARD, for Catalyzed Reporter Deposition — is an enzyme-mediated detection method that utilizes the catalytic activity of horseradish peroxidase (HRP) to generate high-density labeling of a target protein or nucleic acid sequence in situ. Useful to enhance ex vivo surface labeling of cells, the TSA method has been reported to increase the sensitivity by up to 100-fold compared to conventional avidin–biotinylated enzyme complex (ABC) procedures. Also, TSA can be combined with several of our other important technologies, including primary and secondary antibodies, avidin and lectin conjugate, enzyme-labeled fluorescence, cytoskeletal stains, organelle probes, cell tracers and proliferation markers and receptor probes, to enhance cell labeling. These options of improving cell detection should be pursued, since reliable and consistent detection simultaneously in both the blue channel and the red channel will allow for many new and interesting biological experiments to be performed.

Design of a system to allow for investigation of potential approaches to enhance the detection sensitivity of the blue channel as well as improve the fluorescence signal from the labeled cells has already begun. Since it is obviously preferable to be able to experiment with different approaches without using an animal, design emphasis has been placed on artificially duplicating blood flow and light attenuation present during data acquisition from a live animal. The present design involves using a syringe, syringe pump, a rectangular block of PDMS with a microchannel conduit, and samples of animal skin and fur. Blood containing fluorescently labeled cells will be placed into a syringe, and the syringe placed into the syringe pump. The microchannel in the rectangular block of PDMS, approximately 30 microns in diameter, will simulate the artery of the animal from which data is acquired. The PDMS block will be placed on the sample stage with the microchannel in the field-of-view of the microscope objective. A microscope slide

will be between the PDMS and the microscope objective (to help simulate data acquisition under actual conditions). The outlet port of the syringe will be connected to the PDMS block via plastic tubing. The syringe pump will provide for flow of the blood through the microchannel in the PDMS. The samples of animal skin and fur, to be placed on the surface of the block of PDMS in contact with the microscope slide, will be used to duplicate the absorption and scattering experienced by the laser light and fluorescence during acquisition of data. Note that attenuation of light and autofluorescence by the PDMS was measured to be minimal. Attenuation was only approximately 3% and the increase in dc offset of the PMT voltage signal due to autofluorescence was only 0.35 volts (for both excitation wavelengths of the two-slit, two-color system).

It is also possible to enhance the performance of the blue channel, as well as the red channel, through software. Presently, an averaging filter is being used on the raw (acquired) data to produce the voltage waveforms that are analyzed for cell signal. The contribution of high frequency noise components to the waveform is mitigated by averaging the raw data. However, the mid and low frequency components, where signal is present, are effected as well. Therefore, other software filters, such as Butterworth, Chebyshev Type I, Chebyshev Type II, Elliptic, Bessel, etc., which can process the raw data in other ways, such as by Fourier transforming and attenuating as a function of frequency, should be investigated. In addition, several filters can be used together, and it is possible to create custom-made filters as well, offering a myriad of ways to transform the raw data. For the two-slit, two-color system, in conjunction with these filters, cross-correlating the data can be performed. For example, one can convolve the data between the two channels, analyze the resulting waveform, and then reconstruct the waveforms of each channel based on the results. As with the filtering of the data, a multitude of options exist for performing this cross-correlating task, and virtually all warrant investigation. Cross-correlation of the data is especially applicable when the device is operated in the two-slit, single-color mode, since then the excitation and detection process for each slit are theoretically identical.

After modification of the systems is completed, system performance characterization concerning detection of particles and cells in vivo should be performed. Establishing detection limits of both devices is advantageous in helping to interpret the

acquired data as well as in determining if further changes and modifications to the systems should take place . One characterization test, discussed above, has already been performed on the single-slit, single-color system involving establishing the lowest number of labeled cells (DiD labeled LNCaP cells) that need to be present in the animal's body in order for cell detection to take place in a reasonable time period (approximately 10 minutes). The characterization tests recommended to establish the limits of in vivo detectability of labeled cells by the cytometers are:

1) Establish the minimum number of microspheres of different MESF numbers that need to be injected for reliable detection by the systems over a 10-minute period. Beginning with microspheres of large (and known) MESF value, inject the fluorescent microspheres (approximately 6 microns in diameter) into an anesthetized mouse at quantities ranging from 10^2 to 10^5 microspheres. (The MESF rating is the number of molecules of the fluorochrome being used to label the microsphere that yield the same fluorescent intensity as each microsphere. For example, one microsphere having an MESF value of 67451 fluoresces with the same intensity as a cell labeled with 67451 molecules of the same fluorochrome labeling the microsphere.) Injection of 10^2 microspheres corresponds (in a mouse) to approximately 1 microsphere per 10^4 circulating leukocytes, and injection of 10^5 microspheres corresponds to approximately 1 microsphere per 10 circulating leukocytes. Injections should be performed using a 100 μl saline solution. The devices will count the number of microspheres they can detect in vivo over a 10-minute period, determining the average counts per minute detected by each system for each quantity injected of each MESF-value microsphere used. (Fluorescent microspheres might be needed to help answer various circulatory questions in which one would not want to use fluorescently tagged tumor cells.) Repeat this process for microspheres of smaller MESF value until no detection is occurring at the highest injection quantity. Thus, for each system, this will yield a table of average counts per minute for each injection quantity of each MESF-valued micosphere used. Note that two different types of

fluorochromes will have to be used, one for excitation by the 473 nm laser, and the other for excitation by the 632 nm lasers.

2) Establish the minimum number of cells of different fluorescent sources that need to be injected for reliable detection by the systems over a 10-minute period. For this test, prepare and isolate cells of different fluorescent sources, such as LNCaP cells labeled with DiD, LNCaP cells labeled with DiO, and MHCIIGFP$^+$ cells. For each labeled group of cells, inject them into a separate mouse beginning with 10^5 cells. (For the red channel, one can begin with 10^3 cells, since it has already been established that as few as 10^3 cells can be detected.) For each fluorescent source, determine the average number of cells detected per minute by each system. If cells are detected, repeat the experiment with fresh mice and reduce the number of cells injected by one order of magnitude.

3) Establish that the correct ratio of two labeled cell populations can be recovered with the two-slit, two-color system. To determine this system parameter, prepare mixtures of different ratios of two groups of cells, each group having a different fluorescent source, one of which is excited by the 473 nm laser and the other of which is excited by the 632 nm laser. For each mixture, inject them into anesthetized mouse (i.e. inject each mouse with one mixture that has a unique ratio of the two different fluorescent sources). The ratio obtained from in vivo measurements will be compared with the known injected mixture ratio to determine the smallest ratio of two labeled cell populations that the two-slit, two-color instrument can accurately discern.

4) Establish the lowest concentration of fluorescent antibodies needed for intravenous injection. This will be accomplished by injecting unlabeled T-cells into an anesthetized mouse, followed by injection of Cy5.5-conjugated anti-CD4 antibodies. The antibody mixtures injected will vary in antibody concentration from 0.1 to 1.0 mg/kg body weight. In vivo measurements from

both cytometers will follow. Repeat the experiment for a fluorochrome that is excited by the 473 nm laser.

Further modification of the devices will need to take place if the above system performance tests do not yield satisfactory results. Once the devices are functioning as desired, further data acquisition can begin. One of the experiments envisioned for the two-slit, two-color system is determination of the subpopulations of white blood cells that are expressing the GFP gene associated with the MHCII cells in the circulation of a C57LB6 strain mouse. For this experiment, antibodies which bind specifically to one kind of cell and are conjugated with a fluorochrome that can be excited by the 632 nm laser but not the 473 nm laser, such as cychrome, are injected into the C57LB6 strain mouse containing the MHCIIGFP$^+$ cells, which are excited by the 473 nm laser. If the type of cell to which the antibody will bind is not present, then no signal will be detected in the 632 nm channel, since the autofluorescence signal will be much stronger than the signal from one fluorochrome molecule flowing freely in the blood. If the type of cell to which the antibody will bind is present, then the proportion of this type of cell comprising the MHCIIGFP$^+$ cells can be determined by taking the ratio of the cell count in the 632 nm channel to the cell count in the 473 nm channel.

Another experiment recommended involves the detection of apoptotic cells. For this experiment, MLL prostate cancer cells will be incubated in a chemical called camptothecin that will induce apoptosis of these cells. When apoptosis of these cells does occur, phosphatidylserine, a phospholipid on the cell membrane that normally points inward, will be redirected outward, exposing it for binding. A molecule called Annexin-V, which binds to this exposed protein, and is conjugated to the fluorochrome Alexafluor-647, which can be excited by the 632 nm laser, will be incubated with the apoptotic MLL cells to label them. The labeled cells will then be injected into a mouse, and the mouse placed on the sample stage of the single-slit, single-color system. Detection of the labeled cells will prove that apoptotic cells can be detected in vivo, setting the stage for future experiments involving the study of effectiveness of pharmaceutical drugs to annihilate disease-related cells.

A third experiment recommended involves using circulating tumor cell count for monitoring antitumor treatment outcomes. For this experiment, LNCaP cells will be injected into the prostatic capsule of six to eight week old male SCID mice following a transverse incision in the lower abdomen. In approximately six weeks, LN metastases will occur in approximately 20% to 40% of the mice, with tumor volumes of 30 to 50 mm^3 being reached. In vivo flow measurement over the six week period will be performed twice a week following implantation of the LNCaP cells using Cy5.5-labeled PEQ226.5 to detect the number of circulating LNCaP cells. After this six week period of tumor growth and data collection, cancer treatment (PDT) will be performed on the tumor-bearing mice. Again, like the tumor growth period, in vivo flow cytometry will be performed twice a week using Cy5.5-labeled PEQ226.5 to detect the number of circulating LNCaP cells. Three weeks after completion of the cancer treatment, when the treatment has had enough time to completely manifest its results, the mice will be sacrificed by carbon monoxide asphyxiation, at which point tumor volume will be measured and metastatic colonies in the lymph nodes will be quantified. All the data will then be analyzed. The in vivo flow cytometry data obtained during the six-week tumor growth period will provide information on the correlation between circulating tumor cell count and success of the metastasis process. The data acquired during and after PDT treatment will be used to try to determine if the circulating tumor cell (CTC) count is representative of metastatic potential of the tumor, if the CTC count is indicative of tumor burden, and what correlation exists between CTC count and the animal's response to malignant tumor therapy. Note that a possible variation on this experiment is the growth of a tumor within the mouse whose cells are expressing the EGFP gene. This will preclude the need for labeling of the tumor cells with a fluorochrome.

Thus, in conclusion, the ability to monitor circulating cells in vivo in a quantitative way offers a number of advantages. One is able to observe the cell population of interest in its native environment, free of artifacts that can be potentially introduced by cell isolation and processing procedures required to perform conventional flow cytometry measurements. In addition, one can follow the same cell population continuously and over long periods of time to examine the dynamic changes in the circulation of different types of cells. The in vivo flow cytometers offer a clean and

efficient way of counting cells in vivo, and are powerful research tools that can provide new insights in studies of animal models of disease. Ultimately, it is also desired to apply in vivo flow cytometry directly to humans, for the diagnosis of disease and the monitoring of treatment. Locations such as oral mucosa, nailfold, and sclera are areas of the body where superficial blood vessels can be found from which cell count data can be acquired. The major limitation of the technique in application to humans is the need to inject a fluorescent probe, few of which are approved for human medical use. Continuing probe development will be required to make circulating cell count measurements a reality in human patients.

References

1. Brandt BH, Schmidt H, de Angelis G, Zanker KS. Predictive laboratory diagnostics in oncology utilizing blood-borne cancer cells ± current best practice and unmet needs. Cancer Letters 162, S11-16 (2001).
2. Personal communication with MIT flow cytometry lab.
3. Dubey P, Su H, Adonai N, et al. Quantitative imaging of the T cell antitumor response by positron-emission tomography. *Proc Natl Acad Sci U S A* 2003; 100: 1232-7.
4. Dodd SJ, Williams M, Suhan JP, Williams DS, Koretsky AP, Ho C. Detection of single mammalian cells by high-resolution magnetic resonance imaging. *Biophys J* 1999; 76: 103-9.
5. Naumov GN, Wilson SM, MacDonald IC, et al. Cellular expression of green fluorescent protein, coupled with high-resolution in vivo videomicroscopy, to monitor steps in tumor metastasis. *J Cell Sci* 1999; 112: 1835-42.
6. Wang W, Wyckoff JB, Frohlich VC, et al. Single cell behavior in metastatic primary mammary tumors correlated with gene expression patterns revealed by molecular profiling. *Cancer Res* 2002; 62: 6278-88.
7. Sweeney TJ, Mailander V, Tucker AA, et al. Visualizing the kinetics of tumorcell clearance in living animals. *Proc Natl Acad Sci U S A* 1999; 96: 12044-9.
8. Padera TP, Kadambi A, di Tomaso E, et al. Lymphatic metastasis in the absence of functional intratumor lymphatics. *Science* 2002; 296: 1883-6.
9. Novak J, Georgakoudi I, Wei X, Prossin A, Lin CP. In vivo flow cytometer for real-time detection and quantification of circulating cells. *Opt Lett* 2004; 29: 77-9.
10. Chen Z, Milner T, Srinivas S, et al. Noninvasive imaging of in vivo blood flow velocity using optical Doppler tomography. *Opt Lett* 1997; 22: 1119-21.
11. Georgakoudi I, Solban N, Novak J, Rice W, Wei X, Hasan T, Lin CP. In vivo flow cytometry: A new method for enumerating circulating cancer cells. In press in Cancer Research, August 2004.

```
 1  startingtime = input('Enter Start Time  ');
 2
 3  [fidd,messagen] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\Test ↙
    s\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_timedataplot_95','r','l');
 4  [N,count] = fread(fidd,inf,'float32');
 5  fclose(fidd);
 6
 7  maxtime = N(length(N));
 8  mintime = N(1);
 9  length(N);
10  timeincrement = (N(2) - N(1));
11  %pause
12
13  while (startingtime > maxtime)|(startingtime < mintime);
14      if (startingtime > maxtime);
15      disp('Starting time specified exceeds starting time available from file--Maximum ↙
    start time available is ');
16      maxtime
17      startingtime = input('Enter Start Time  ');
18      end
19      if (startingtime < mintime);
20      disp('Starting time specified is less than starting time available from file--Min↙
    imum start time available is ');
21      mintime
22      startingtime = input('Enter Start Time  ');
23      end
24  end
25
26  startingtime;
27  %pause
28  timeindicesallowed = find(startingtime<=N);
29  N = N(timeindicesallowed(:));
30  lengthofN = length(N);
31
32  [fid,messagem] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\Tests ↙
    \DiD50KHzsamplingnoLPF024ND260microwattstrace2av_rawdataplot_95','r','l');
33  %[M,count] = fread(fid,inf,'float64');
34  [M,count] = fread(fid,inf,'float32');
35  fclose(fid)
36  length(M);
37  %pause
38  M = M(timeindicesallowed(:));
39
40  [fiddddd,messagep] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\T↙
    ests\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_boundarytotal','r','l');
41  %[P,count] = fread(fiddddd,inf,'float64');
42  [P,count] = fread(fiddddd,inf,'float32');
43  fclose(fiddddd)
44  P = P(timeindicesallowed(:));
```

```
45
46  [fiddddd1,messagef] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\ ↙
    Tests\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_firstnumtotal','r','l');
47  %[F,count] = fread(fiddddd1,inf,'float64');
48  [F,count] = fread(fiddddd1,inf,'float32');
49  fclose(fiddddd1)
50  F = F(timeindicesallowed(:));
51
52  timeintervallength = 0.2;
53  %timeincrement = input('Enter Time Increment:  ');
54  n = round(timeintervallength/timeincrement);
55
56  if lengthofN < n
57      n = lengthofN;
58  end
59
60  ntotal = 0;
61
62  while ntotal < lengthofN
63  timeintervallength = input('Enter Time Interval Length:  ');
64  if length(timeintervallength) == 0
65      n = n;
66  elseif timeintervallength < timeincrement
67      n = 1;
68  else
69      n = round(timeintervallength/timeincrement);
70  end
71  n;
72  if ntotal + n + 200 > lengthofN
73      n = lengthofN - ntotal;
74  end
75
76  clf
77
78  MM = M(ntotal+1:ntotal+n);
79
80  NN = N(ntotal+1:ntotal+n);
81
82  %NN(1)
83  %NN(length(NN))
84
85  %plot(NN,MM,'b-')
86  %axis([NN(1) NN(length(NN)) 0.6 1])
87  %hold on
88
89  PP = P(ntotal+1:ntotal+n);
90
91  NN = N(ntotal+1:ntotal+n);
92
```

```
 93  %plot(NN,PP,'k-');
 94  hold on
 95
 96  FF = F(ntotal+1:ntotal+n);
 97
 98  NN = N(ntotal+1:ntotal+n);
 99
100  %plot(NN,FF,'r-');
101  hold on
102
103  [gfidddd,messager] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\T↙
     ests\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_timemoundplot_95','r','l');
104  [S,count] = fread(gfidddd,inf,'float32');
105  fclose(gfidddd);
106  timeindices = find((NN(1)<=S) & (NN(n)>=S));
107  S = S(timeindices(:));
108
109  [gfiddd,messageq] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\Te↙
     sts\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_moundplot_95','r','l');
110  %[T,count] = fread(gfiddd,inf,'float64');
111  [T,count] = fread(gfiddd,inf,'float32');
112  fclose(gfiddd);
113  T = T(timeindices(:));
114
115  %plot(S,T,'k.');
116  %hold on
117
118  [ggfidddd,messager] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\↙
     Tests\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_signaltimemoundplot_95','r','l'↙
     );
119  [U,count] = fread(ggfidddd,inf,'float32');
120  fclose(ggfidddd);
121  timeindices = find((NN(1)<=U) & (NN(n)>=U));
122  U = U(timeindices(:));
123
124  [ggfiddd,messageq] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\T↙
     ests\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_signalmoundplot_95','r','l');
125  %[V,count] = fread(ggfiddd,inf,'float64');
126  [V,count] = fread(ggfiddd,inf,'float32');
127  fclose(ggfiddd);
128  V = V(timeindices(:));
129
130  %plot(U,V,'y.');
131  %hold on
132
133  [fidddd,messager] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\Te↙
     sts\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_peaktimeplot_95','r','l');
134  [R,count] = fread(fidddd,inf,'float32');
135  fclose(fidddd);
```

```
136  timeindices = find((NN(1)<=R) & (NN(n)>=R));
137  R = R(timeindices(:));
138
139  [fiddd,messageq] = fopen('C:\Documents and Settings\John Novak\My Documents\Novak\Tes↙
     ts\DiD50KHzsamplingnoLPF024ND260microwattstrace2av_peakdataplot_95','r','l');
140  %[Q,count] = fread(fiddd,inf,'float64');
141  [Q,count] = fread(fiddd,inf,'float32');
142  fclose(fiddd);
143  Q = Q(timeindices(:));
144
145  %plot(R,Q,'r.');
146  %hold
147
148  NN(1)
149  NN(length(NN))
150
151  plot(NN,MM,'b-')
152  %axis([NN(1) NN(length(NN)) 0 10]);
153  hold on
154
155  plot(NN,PP,'k-');
156  %axis([NN(1) NN(length(NN)) 0 10]);
157  hold on
158
159  plot(NN,FF,'r-');
160  %axis([NN(1) NN(length(NN)) 0 10]);
161  hold on
162
163  plot(S,T,'k.');
164  %axis([NN(1) NN(length(NN)) 0 10]);
165  hold on
166
167  plot(U,V,'y.');
168  %axis([NN(1) NN(length(NN)) 0 10]);
169  hold on
170
171  plot(R,Q,'r.');
172  %axis([NN(1) NN(length(NN)) 0 10]);
173  hold
174
175  ntotal = ntotal + n;
176  end
```

```
function [data_smf]=binarysmoothingfile(data_filename,smdata_filename,starttime,endtime,nsm,sizeofint,nofint)
%data_smf is the smoothed data vector
%data_filename is the name of the binary file to be smoothed
%smdata_filename is the name of the binary file where the smoothed data
%should be saved
%starttime is the time where smoothing starts
%endtime is the time when smoothing ends
%nsm is the width of the moving window average
%the product of sizeofint and nofint determines how many points will be
%smoothed at once

%pause
%format long
%clear
%M = dlmread('DID--RBC--3.00 volts PMT--031003.txt','\t',[1 0 100010 1]);
%starttime = 0.0;
%endtime = 120;
%nsm = 101; %nsm must be an odd integer number
%sizeofint = 1000000; %sizeofint must be greater than or equal to nsm and must be an integer
%nofint = 12; %number of sizeofint one wants to analyze for each j
copyfile(data_filename,smdata_filename);
clear data_sm
%pause
%[fid,message] = fopen('DID--NALM6--40 min after inj--032703--Artery.dcf','r','l');
%if j == 1
%[fid,message] = fopen('DID--RBC--3.00 volts PMT--031003--Vein3.dcf','r','l');
%[fidd,message] = fopen('DID--RBC--3.00 volts PMT--031003--smoothedVein3.dcf','w','l');
%Header = fread(fid,120,'uchar');
%fwrite(fidd,Header,'uchar');
%fclose(fidd);
%fclose(fid);
%end
[fid,message] = fopen(data_filename,'r','l');
%
Magic = fread(fid,1,'uint32');
Version = fread(fid,1,'uint32');
i = 0;
feof(fid) = 0;
while feof(fid)~=1
    tmp = fread(fid,1,'uint32');
    if feof(fid)
        break
    end
    i = i + 1;
    A(i).Length = fread(fid,1,'uint32');
    A(i).Datatype = fread(fid,1,'uint16');
```

```
47      switch A(i).Datatype;
48          case 2
49              A(i).data_width = 2;
50              A(i).data_format = 'int16';
51          case 3
52              A(i).data_width = 4;
53              A(i).data_format = 'int32';
54          case 4
55              A(i).data_width = 4;
56              A(i).data_format = 'float32';
57          case 5
58              A(i).data_width = 8;
59              A(i).data_format = 'float64';
60          case 17
61              A(i).data_width = 1;
62              A(i).data_format = 'int8';
63          otherwise
64              fclose(fid);
65              error('Unspecified data type');
66      end
67      A(i).data_width;
68      %pause
69      A(i).TimeAxis = fread(fid,1,'uint16');
70      A(i).AssignedTimeAxis = fread(fid,1,'int32');
71      A(i).starttime = fread(fid,1,'float64');
72      A(i).TimeIncrement = fread(fid,1,'float64');
73      if starttime > (round((A(i).Length - 112)/A(i).data_width)-1)*A(i).TimeIncrement
74              break %from while command--starttime exceeds endtime available from file ↙
    read
75      end
76      A(i).starttime = round(starttime/A(i).TimeIncrement) * A(i).TimeIncrement;
77      A(i).scale = fread(fid,1,'float64');
78      A(i).offset = fread(fid,1,'float64');
79      A(i).EngRangeMin = fread(fid,1,'float64');
80      A(i).EngRangeMax = fread(fid,1,'float64');
81      A(i).engunitlength = fread(fid,1,'uint8');
82      A(i).engunit = char(fread(fid, A(i).engunitlength, 'char')');
83      fseek(fid, 15 - A(i).engunitlength, 'cof');
84      A(i).descriptionlength = fread(fid,1,'uint8');
85      A(i).Description = char(fread(fid, A(i).descriptionlength, 'char')');
86      fseek(fid, 31 - A(i).descriptionlength, 'cof');
87      %ftell(fid)
88      fseek(fid, round((starttime/A(i).TimeIncrement)*A(i).data_width), 'cof');
89      A(i).num_entries = round((A(i).Length - 112) / A(i).data_width);
90      A(i).num_entries_available = round(A(i).num_entries - round(starttime/A(i).TimeIn ↙
    crement));
91      totalpoints = round((endtime - starttime)/A(i).TimeIncrement) + 1; %total number ↙
    of points being evaluated
92      if totalpoints > A(i).num_entries_available
```

```
93          %totalpoints
94          %starttime
95          %A(i).num_entries_available
96          %A(i).TimeIncrement
97          %pause
98          disp('Error--endtime exceeds endtime available from file')
99          newendtime = starttime + ((A(i).num_entries_available - 1)*A(i).TimeIncrement ↙
    )
100         A(i).num_entries_final = round(A(i).num_entries_available);
101     elseif totalpoints <= A(i).num_entries_available
102         A(i).num_entries_final = round(totalpoints);
103     end
104     A(i).data = fread(fid, A(i).num_entries_final, A(i).data_format);
105     A(i).time = linspace(A(i).starttime, A(i).starttime + (A(i).TimeIncrement * (A(i) ↙
    .num_entries_final  - 1)), A(i).num_entries_final);
106     feof(fid) = 1;
107     %A
108     %pause
109 end % for while command
110 fclose(fid);
111 %
112 if  starttime > (((A(i).Length - 112)/A(i).data_width)-1)*A(i).TimeIncrement
113      disp('Error--Starting time exceeds time available from file')
114     %break % break from for loop--starttime exceeds endtime available from file read
115 else
116 data = A(1).data(:);
117 [data_sm] = filter(ones(1,nsm)/nsm,1,data);
118 %n = A(i).num_entries;
119 n =  round(A(i).num_entries_final);
120 %pause
121 data_smf(1:n) = data_sm(1:n);
122 [fidd,message] = fopen(smdata_filename,'r+','l');
123 fseek(fidd,120,'bof');
124 switch A(i).Datatype;
125     case 4
126 fwrite(fidd,data_sm(1:n),'float32');
127     case 5
128 fwrite(fidd,data_sm(1:n),'float64');
129 end
130 fclose(fidd);
131 %
132 data_smf = data_smf';
133 end
134 %plot(time,data_smf)
135 %xlabel('Time (sec)');
136 %ylabel('Voltage Value (Volts)');
137 %title('Signal from DID--RBC--3.00 volts PMT--031003--Vein3.dcf--nsm=101--sizeofint=2 ↙
    00')
138 %axis([0 0.2 0 1.6])
```

```
139   %grid on
140   %end %if command
```

```
 1  function [int_threshold, hw_threshold] = binaryreadingcellcounting_new_John(filename,
    root,starttime,endtime,nsm,sizeofint,nofint,...
 2      counter_vector,rmst,vt,w1,wt,stdv_th,slope,intercept)
 3
 4  %filename is the name of the smoothed binary file
 5  %root is the beginning of the filename to which the height,width etc
 6  %information is saved. The ultimate filename will be something like
 7  %root_95.txt or root_90.txt, depending on your percentage for the
 8  %threshold.
 9  %starttime is the time where cell counting starts
10  %endtime is when cell counting ends
11  %nsm is the size of the moving window average used for smoothing the
12  %original data file
13  %the product of sizeofint and nofint determines how many points are read
14  %from the file at a time for cell counting
15  %counter_vector is a vector that determines how the mean and standard
16  %deviation of the noise are calculated. For manual threshold setting
17  %counter_vector=[1]. For manual threshold setting and for mean and std
18  %calculations based on 50% of the points counter_vector=[1 2]. For mean and
19  %std calculation based on 70 90 93 and 95% of the points
20  %counter_vector=[3 4 5 6].
21  %rmst is used only if you do manual threshold setting, but even if you
22  %don't, you should provide some number for it, even if it doesn't really
23  %get used.
24  %vt is the number of standard deviations used for intensity thresholding. 4
25  %is a good number
26  %w1 is used for peak counting. 50 is a good number
27  %wt is a width threshold for peak counting
28  %stdv_th is another threshold used mainly during consideration of closely
29  %spaced peaks. 4 is a good number as well
30
31  %format short e
32  format long
33  %
34  %clear
35  x=size(counter_vector);
36  for counter = counter_vector(1):1:counter_vector(x(1,2))
37      nvnewm = 0;
38      nvnewa50 = 0;
39      nvnewa70 = 0;
40      nvnewa90 = 0;
41      nvnewa93 = 0;
42      nvnewa95 = 0;
43      diff = 0;
44      nevaluated = 0;
45      EOFM = 0;
46      datamatrixcount = 0;
47      signaldatamatrixcount = 0;
48      %M = dlmread('DID--RBC--3.00 volts PMT--031003.txt','\t',[1 0 100010 1]);
```

```
49      %starttime = 0;
50      %endtime = 120;
51      %nsm = 101; %nsm must be an odd integer number
52      %sizeofint = 200; %sizeofint must be greater than or equal to nsm and must be an integer
53      %nofint = 50; %number of sizeofint one wants to analyze for each j
54      m = nofint*sizeofint; %total number of data points being analyzed for each j
55      %initialtotalpoints = round(((endtime - starttime)/1e-05) + 1); %total number of points being evaluated
56
57
58      %jmax = round(totalpoints/n); %total number of points being evaluated = jmax*n
59      j = 0;
60      %for j = 1:jmax
61      while EOFM == 0;
62          j = j + 1;
63          n = m + diff;
64          clear time
65          clear data_smf
66
67          %[fid,message] = fopen('DID--NALM6--40 min after inj--032703--Artery.dcf','r','l');
68          [fid,message] = fopen(filename,'r','l');
69          %
70          Magic = fread(fid,1,'uint32');
71          Version = fread(fid,1,'uint32');
72          i = 0;
73          feof(fid) = 0;
74          while feof(fid)~=1
75              tmp = fread(fid,1,'uint32');
76              if feof(fid)
77                  break
78              end
79              i = i + 1;
80              A(i).Length = fread(fid,1,'uint32');
81              A(i).Datatype = fread(fid,1,'uint16');
82              switch A(i).Datatype;
83                  case 2
84                      A(i).data_width = 2;
85                      A(i).data_format = 'int16';
86                  case 3
87                      A(i).data_width = 4;
88                      A(i).data_format = 'int32';
89                  case 4
90                      A(i).data_width = 4;
91                      A(i).data_format = 'float32';
92                  case 5
93                      A(i).data_width = 8;
94                      A(i).data_format = 'float64';
```

```
 95                 case 17
 96                     A(i).data_width = 1;
 97                     A(i).data_format = 'int8';
 98                 otherwise
 99                     fclose(fid);
100                     error('Unspecified data type');
101             end
102             %A(i).data_width
103             %A(i).num_entries = (A(i).Length - 112) / A(i).data_width;
104             totalnum_entries = (A(i).Length - 112) / A(i).data_width;
105             A(i).TimeAxis = fread(fid,1,'uint16');
106             A(i).AssignedTimeAxis = fread(fid,1,'int32');
107             A(i).starttime = fread(fid,1,'float64');
108             A(i).TimeIncrement = fread(fid,1,'float64');
109              if (j == 1) & (starttime > (round((A(i).Length - 112)/A(i).data_width)-1↙
    )*A(i).TimeIncrement)
110                 disp('Error--Starting time exceeds time available from file')
111                 break %from while feof(fid) ~= 1 command--starttime exceeds endtime ↙
    available from file read
112             end
113             totalpoints = round((endtime - starttime)/A(i).TimeIncrement) + 1; %total↙
     number of points being evaluated
114             if (j == 1) & (totalpoints > (totalnum_entries - round(starttime/A(i).Tim↙
    eIncrement)))
115                 totalpoints = totalnum_entries - round(starttime/A(i).TimeIncrement);
116                 disp('Error--endtime exceeds endtime available from file')
117                 newendtime = starttime + (totalpoints - 1)*A(i).TimeIncrement
118                 endtime = newendtime;
119             elseif (j == 1) & (totalpoints <= (totalnum_entries - round(starttime/A(i↙
    ).TimeIncrement)))
120                 totalpoints = totalpoints;
121             end
122             %
123               if (n + nevaluated + 200) > totalpoints
124                   n = totalpoints - nevaluated;
125             end
126             %
127             %A(i).starttime = starttime + A(i).starttime+(j-1)*n*A(i).TimeIncrement;
128             A(i).starttime = ((round(starttime/A(i).TimeIncrement))*A(i).TimeIncremen↙
    t) + (nevaluated*A(i).TimeIncrement);
129             A(i).scale = fread(fid,1,'float64');
130             A(i).offset = fread(fid,1,'float64');
131             A(i).EngRangeMin = fread(fid,1,'float64');
132             A(i).EngRangeMax = fread(fid,1,'float64');
133             A(i).engunitlength = fread(fid,1,'uint8');
134             A(i).engunit = char(fread(fid, A(i).engunitlength, 'char')');
135             fseek(fid, 15 - A(i).engunitlength, 'cof');
136             A(i).descriptionlength = fread(fid,1,'uint8');
137             A(i).Description = char(fread(fid, A(i).descriptionlength, 'char')');
```

```
138             fseek(fid, 31 - A(i).descriptionlength, 'cof');
139             ftell(fid);
140             fseek(fid,(round(starttime/A(i).TimeIncrement)+round(nevaluated))*A(i).da↙
    ta_width,'cof');
141             ftell(fid);
142             A(i).num_entries_available = round(totalnum_entries - round(starttime/A(i↙
    ).TimeIncrement) - nevaluated);
143             if n > A(i).num_entries_available
144                 n = A(i).num_entries_available;
145             elseif n <= A(i).num_entries_available
146                 n = n;
147             end
148             A(i).data = fread(fid, n, A(i).data_format);
149             A(i).time = linspace(A(i).starttime, A(i).starttime + (A(i).TimeIncrement↙
     * (n - 1)), n);
150             feof(fid) = 1;
151         end %while feof(fid)~=1
152         fclose(fid);
153         %
154         if (j == 1) & (starttime > (round((A(i).Length - 112)/A(i).data_width)-1)*A(i↙
    ).TimeIncrement)
155                 break %from while EOFM == 0 command--starttime exceeds endtime avail↙
    able from file read
156         end
157         %totalpoints;
158         %n;
159         %if (length(A(1).data) < n + ((nsm-1)/2))
160         %time(((j-1)*n)+1:((j-1)*n)+length(A(1).data)-((nsm-1)/2)) = A(1).time(1:leng↙
    th(A(1).data)-((nsm-1)/2));
161         %time((nevaluated)+1:(nevaluated)+length(A(1).data)-((nsm-1)/2)) = A(1).time(↙
    1:length(A(1).data)-((nsm-1)/2));
162         %time(1:length(A(1).data)-((nsm-1)/2)) =  A(1).time(1:length(A(1).data)-((nsm↙
    -1)/2));
163         %else
164         %time((nevaluated)+1:nevaluated+n) = A(1).time(1:n);
165         %time(1:A(i).num_entries) = A(1).time(1:A(i).num_entries);
166         time(1:n) = A(1).time(1:n);
167         %end
168         %
169         data_smf = A(i).data(:);
170         %
171
172         %if (j == 1) & (counter == 1)
173         %   [fid,message] = fopen('DID--RBC--3.00 volts PMT--031103--timesme:\Irene\0↙
    50803\control-artery1_.dcf','w','l');
174         %   totaltime = linspace(A(i).starttime, A(i).starttime + (A(i).TimeIncrement↙
     * (totalpoints - 1)), totalpoints);
175         %   fwrite(fid,totaltime','float64');
176         %   fclose(fid);
```

```
177          %end
178          %rmst = 0.375;
179          %vt = 4;
180          %w1 = 50;
181          %wt = 0.000;
182          %stdv_th = 4;
183          %[rms,meanv,stdv,nv,index,t,dt,height,width] = analysis3(data_smf,time,rmst,v↙
     t,w1,wt);
184
185          %size(data_smf)
186          %size(time)
187          %rmst
188          %vt
189          %w1
190          %wt
191          %stdv_th
192          %totalpoints
193          %nevaluated
194          %nsm
195          %counter
196          %slope
197          %intercept
198
199          [rms,meanv,stdv,n,nv,index,t,dt,height,width,EOFM,diff,nevaluated,datamatrixt↙
     otal,timematrixtotal,boundary,firstnum,absheight,signaldatamatrixtotal,signaltimematr↙
     ixtotal] = analysis8_newmod54_linear_MF_Scope(data_smf,time,rmst,vt,w1,wt,stdv_th,tot↙
     alpoints,nevaluated,nsm,counter,slope,intercept);
200
201           if j == 1
202             data_smftotal = data_smf(1:n);
203             timetotal = time(1:n);
204         elseif j > 1
205             data_smftotal(length(data_smftotal) + 1:length(data_smftotal) + n) = data_↙
     smf(1:n);
206             timetotal(length(timetotal) + 1:length(timetotal) + n) = time(1:n);
207         end
208
209          if j == 1
210              boundarytotal(1:n) = boundary;
211          elseif j > 1
212              boundarytotal(length(boundarytotal) + 1:length(boundarytotal) + n) = boun↙
     dary;
213          end
214
215          if j == 1
216              firstnumtotal(1:n) = firstnum;
217          elseif j > 1
218              firstnumtotal(length(firstnumtotal) + 1:length(firstnumtotal) + n) = firs↙
     tnum;
```

```
219         end
220
221          datamatrixcount = datamatrixcount + 1;
222         if datamatrixcount == 1
223                     datamatrixtotaltotal = datamatrixtotal(:);
224                     timematrixtotaltotal = timematrixtotal(:);
225                     clear datamatrixtotal timematrixtotal;
226           elseif datamatrixcount > 1
227                     datamatrixtotaltotal(length(datamatrixtotaltotal) + 1:length(datam↙
    atrixtotaltotal) + length(datamatrixtotal)) = datamatrixtotal(:);
228                     timematrixtotaltotal(length(timematrixtotaltotal) + 1:length(timem↙
    atrixtotaltotal) + length(timematrixtotal)) = timematrixtotal(:);
229                     clear datamatrixtotal timematrixtotal
230          end
231
232         if nv > 0
233             if counter == 1
234                 widthm(1+nvnewm:nvnewm+nv,1) = width(1:nv);
235                 heightm(1+nvnewm:nvnewm+nv,1) = height(1:nv);
236                 tm(1+nvnewm:nvnewm+nv,1) = t(1:nv);
237                 dtm(1+nvnewm:nvnewm+nv,1) = dt(1:nv);
238                 indexm(1+nvnewm:nvnewm+nv,1) = index(1:nv);
239                 nvnewm = nvnewm + nv;
240                 %hold on;
241                 %grid on;
242                 %axis([0 0.05 0 0.4]);
243                 %plot(width,height,'bd')
244             elseif counter == 2
245                 widtha50(1+nvnewa50:nvnewa50+nv,1) = width(1:nv);
246                 heighta50(1+nvnewa50:nvnewa50+nv,1) = height(1:nv);
247                 ta50(1+nvnewa50:nvnewa50+nv,1) = t(1:nv);
248                 dta50(1+nvnewa50:nvnewa50+nv,1) = dt(1:nv);
249                 indexa50(1+nvnewa50:nvnewa50+nv,1) = index(1:nv);
250                 nvnewa50 = nvnewa50 + nv;
251             elseif counter == 3
252                 widtha70(1+nvnewa70:nvnewa70+nv,1) = width(1:nv);
253                 heighta70(1+nvnewa70:nvnewa70+nv,1) = height(1:nv);
254                 ta70(1+nvnewa70:nvnewa70+nv,1) = t(1:nv);
255                 dta70(1+nvnewa70:nvnewa70+nv,1) = dt(1:nv);
256                 indexa70(1+nvnewa70:nvnewa70+nv,1) = index(1:nv);
257                 nvnewa70 = nvnewa70 + nv;
258             elseif counter == 4
259                 widtha90(1+nvnewa90:nvnewa90+nv,1) = width(1:nv);
260                 heighta90(1+nvnewa90:nvnewa90+nv,1) = height(1:nv);
261                 ta90(1+nvnewa90:nvnewa90+nv,1) = t(1:nv);
262                 dta90(1+nvnewa90:nvnewa90+nv,1) = dt(1:nv);
263                 indexa90(1+nvnewa90:nvnewa90+nv,1) = index(1:nv);
264                 nvnewa90 = nvnewa90 + nv;
265             elseif counter == 5
```

```
266                 widtha93(1+nvnewa93:nvnewa93+nv,1) = width(1:nv);
267                 heighta93(1+nvnewa93:nvnewa93+nv,1) = height(1:nv);
268                 ta93(1+nvnewa93:nvnewa93+nv,1) = t(1:nv);
269                 dta93(1+nvnewa93:nvnewa93+nv,1) = dt(1:nv);
270                 indexa93(1+nvnewa93:nvnewa93+nv,1) = index(1:nv);
271                 nvnewa93 = nvnewa93 + nv;
272             elseif counter == 6
273                 nv;
274                 width;
275                 nvnewa95;
276                     widtha95(1+nvnewa95:nvnewa95+nv,1) = width(1:nv);
277                     heighta95(1+nvnewa95:nvnewa95+nv,1) = height(1:nv);
278                     absheighta95(1+nvnewa95:nvnewa95+nv,1) = absheight(1:nv);
279                     ta95(1+nvnewa95:nvnewa95+nv,1) = t(1:nv);
280                     dta95(1+nvnewa95:nvnewa95+nv,1) = dt(1:nv);
281                     indexa95(1+nvnewa95:nvnewa95+nv,1) = index(1:nv);
282                     nvnewa95 = nvnewa95 + nv;
283                     %datamatrixcount = datamatrixcount + 1;
284                     signaldatamatrixcount = signaldatamatrixcount + 1;
285                     %if datamatrixcount == 1
286                         %datamatrixtotaltotal = datamatrixtotal(:);
287                         %timematrixtotaltotal = timematrixtotal(:);
288                         %clear datamatrixtotal timematrixtotal;
289                         %elseif datamatrixcount > 1
290                         %datamatrixtotaltotal(length(datamatrixtotaltotal) + 1:length↙
(datamatrixtotaltotal) + length(datamatrixtotal)) = datamatrixtotal(:);
291                         %timematrixtotaltotal(length(timematrixtotaltotal) + 1:length↙
(timematrixtotaltotal) + length(timematrixtotal)) = timematrixtotal(:);
292                         %clear datamatrixtotal timematrixtotal
293                         %end
294                      if signaldatamatrixcount == 1
295                         signaldatamatrixtotaltotal = signaldatamatrixtotal(:);
296                         signaltimematrixtotaltotal = signaltimematrixtotal(:);
297                         clear signaldatamatrixtotal signaltimematrixtotal;
298                     elseif signaldatamatrixcount > 1
299                         signaldatamatrixtotaltotal(length(signaldatamatrixtotaltotal)↙
 + 1:length(signaldatamatrixtotaltotal) + length(signaldatamatrixtotal)) = signaldata↙
matrixtotal(:);
300                         signaltimematrixtotaltotal(length(signaltimematrixtotaltotal)↙
 + 1:length(signaltimematrixtotaltotal) + length(signaltimematrixtotal)) = signaltime↙
matrixtotal(:);
301                         clear signaldatamatrixtotal signaltimematrixtotal
302                     end
303             end %if counter == 1
304         end %if nv > 0
305         %if (j == 1) & (starttime > (((A(i).Length - 112)/A(i).data_width) - n - ((ns↙
m-1)/2))*A(i).TimeIncrement) & (starttime <= (((A(i).Length - 112)/A(i).data_width)-1↙
)*A(i).TimeIncrement)
306         % disp('Note: End of data reached from file read before specified endtime')
```

```
307         % break %break from for loop with jmax--end of data reached from file read be↙
    fore specified endtime
308         %end
309         %if (j ~= 1) & (length(A(1).data) <= n + ((nsm-1)/2))
310         % disp('Note: End of data reached from file read before specified endtime')
311         % break %break from for loop with jmax--end of data reached from file read be↙
    fore specified endtime
312         %end
313     end %while EOFM == 0;
314     %plot(width,height,'bd')
315     if (j == 1) & (starttime > (round((A(i).Length - 112)/A(i).data_width)-1)*A(i).Ti↙
    meIncrement)
316                 break %from for counter = counter_vector(1):1:counter_vector(x(1,2)) ↙
    command--starttime exceeds endtime available from file read
317     end
318         x1=0.06:0.12:(84*0.12-0.06);
319         x1=x1';
320         x2=0.001:0.002:(200*0.002-0.001);
321         x2=x2';
322
323     if counter == 1
324         cellnum_m=0;
325         for k=1:84
326             a=find(((k-1)*0.12 < heightm) & (heightm< k*0.12) & (heightm-slope*widthm↙
    -intercept>0.0));
327             num=size(a);
328             histo_height_m(k,1)=num(1,1)*60/(endtime-starttime);
329             cellnum_m=cellnum_m+histo_height_m(k,1);
330         end
331
332         for k=1:200
333             a=find(((k-1)*0.002 < widthm) & (widthm< k*0.002)& (heightm-slope*widthm-↙
    intercept>0.0));
334             num=size(a);
335             histo_width_m(k,1)=num(1,1)*60/(endtime-starttime);
336         end
337
338         filename1=strcat(root,'_m.txt');
339         filename2=strcat(root, '_pkctm.txt');
340         filename3=strcat(root,'_histo_width_m.txt');
341         filename4=strcat(root,'_histo_height_m.txt');
342
343
344         eval(['dlmwrite(''' filename1 ''',[widthm,heightm,tm,dtm,indexm],''\t'')'])
345         eval(['dlmwrite(''' filename2 ''',[nvnewm,cellnum_m],''\t'')'])
346         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_m],''\t'')'])
347         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_m],''\t'')'])
348
349         %save the plottable data
```

```
350         save_plot_variables(filename, heightm, widthm, x1, histo_height_m, x2, histo_
    width_m);
351         int_threshold = nvnewm;
352         hw_threshold = cellnum_m;
353
354     elseif counter == 2
355         cellnum_50=0;
356         for k=1:84
357             a=find(((k-1)*0.12 < heighta50) & (heighta50< k*0.12) & (heighta50-slope*
    widtha50-intercept>0.0));
358             num=size(a);
359             histo_height_50(k,1)=num(1,1)*60/(endtime-starttime);
360             cellnum_50=cellnum_50+histo_height_50(k,1);
361         end
362
363         for k=1:200
364             a=find(((k-1)*0.002 < widtha50) & (widtha50< k*0.002)& (heighta50-slope*w
    idtha50-intercept>0.0));
365             num=size(a);
366             histo_width_50(k,1)=num(1,1)*60/(endtime-starttime);
367         end
368
369         filename1=strcat(root,'_50.txt');
370         filename2=strcat(root, '_pkcta50.txt');
371         filename3=strcat(root,'_histo_width_50.txt');
372         filename4=strcat(root,'_histo_height_50.txt');
373
374
375         eval(['dlmwrite(''' filename1 ''',[widtha50,heighta50,ta50,dta50,indexa50],''
    \t'')'])
376         eval(['dlmwrite(''' filename2 ''',[nvnewa50,cellnum_50],''\t'')'])
377         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_50],''\t'')'])
378         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_50],''\t'')'])
379         %save the plottable data
380         save_plot_variables(filename, heighta50, widtha50, x1, histo_height_50, x2, h
    isto_width_50);
381         int_threshold = nvnewa50;
382         hw_threshold = cellnum_50;
383
384     elseif counter == 3
385         cellnum_70=0;
386         for k=1:84
387             a=find(((k-1)*0.12 < heighta70) & (heighta70< k*0.12) & (heighta70-slope*
    widtha70-intercept>0.0));
388             num=size(a);
389             histo_height_70(k,1)=num(1,1)*60/(endtime-starttime);
390             cellnum_70=cellnum_70+histo_height_70(k,1);
391         end
392
```

```
393         for k=1:200
394             a=find(((k-1)*0.002 < widtha70) & (widtha70< k*0.002)& (heighta70-slope*w↙
    idtha70-intercept>0.0));
395             num=size(a);
396             histo_width_70(k,1)=num(1,1)*60/(endtime-starttime);
397         end
398
399         filename1=strcat(root,'_70.txt');
400         filename2=strcat(root, '_pkcta70.txt');
401         filename3=strcat(root,'_histo_width_70.txt');
402         filename4=strcat(root,'_histo_height_70.txt');
403
404
405         eval(['dlmwrite(''' filename1 ''',[widtha70,heighta70,ta70,dta70,indexa50],''↙
    \t'')'])
406         eval(['dlmwrite(''' filename2 ''',[nvnewa70,cellnum_70],''\t'')'])
407         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_70],''\t'')'])
408         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_70],''\t'')'])
409         %save the plottable data
410         save_plot_variables(filename, heighta70, widtha70, x1, histo_height_70, x2, h↙
    isto_width_70);
411         int_threshold = nvnewa70;
412         hw_threshold = cellnum_70;
413
414     elseif counter == 4
415
416         cellnum_90=0;
417         for k=1:84
418             a=find(((k-1)*0.12 < heighta90) & (heighta90< k*0.12) & (heighta90-slope*↙
    widtha90-intercept>0.0));
419             num=size(a);
420             histo_height_90(k,1)=num(1,1)*60/(endtime-starttime);
421             cellnum_90=cellnum_90+histo_height_90(k,1);
422         end
423
424         for k=1:200
425             a=find(((k-1)*0.002 < widtha90) & (widtha90< k*0.002)& (heighta90-slope*w↙
    idtha90-intercept>0.0));
426             num=size(a);
427             histo_width_90(k,1)=num(1,1)*60/(endtime-starttime);
428         end
429
430         filename1=strcat(root,'_90.txt');
431         filename2=strcat(root, '_pkcta90.txt');
432         filename3=strcat(root,'_histo_width_90.txt');
433         filename4=strcat(root,'_histo_height_90.txt');
434
435
436         eval(['dlmwrite(''' filename1 ''',[widtha90,heighta90,ta90,dta90,indexa90],''↙
```

```
    \t'')'])
437         eval(['dlmwrite(''' filename2 ''',[nvnewa90,cellnum_90],''\t'')'])
438         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_90],''\t'')'])
439         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_90],''\t'')'])
440
441         %save the plottable data
442         save_plot_variables(filename, heighta90, widtha90, x1, histo_height_90, x2, h↙
    isto_width_90);
443         int_threshold = nvnewa90;
444         hw_threshold = cellnum_90;
445
446     elseif counter == 5
447
448         cellnum_93=0;
449         for k=1:84
450             a=find(((k-1)*0.12 < heighta93) & (heighta93< k*0.12) & (heighta93-slope*↙
    widtha93-intercept>0.0));
451             num=size(a);
452             histo_height_93(k,1)=num(1,1)*60/(endtime-starttime);
453             cellnum_93=cellnum_93+histo_height_93(k,1);
454         end
455
456         for k=1:200
457             a=find(((k-1)*0.002 < widtha93) & (widtha93< k*0.002)& (heighta93-slope*w↙
    idtha93-intercept>0.0));
458             num=size(a);
459             histo_width_93(k,1)=num(1,1)*60/(endtime-starttime);
460         end
461
462         filename1=strcat(root,'_93.txt');
463         filename2=strcat(root, '_pkcta93.txt');
464         filename3=strcat(root,'_histo_width_93.txt');
465         filename4=strcat(root,'_histo_height_93.txt');
466
467
468         eval(['dlmwrite(''' filename1 ''',[widtha93,heighta93,ta93,dta93,indexa93],''↙
    \t'')'])
469         eval(['dlmwrite(''' filename2 ''',[nvnewa93,cellnum_93],''\t'')'])
470         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_93],''\t'')'])
471         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_93],''\t'')'])
472         %save the plottable data
473         save_plot_variables(filename, heighta93, widtha93, x1, histo_height_93, x2, h↙
    isto_width_93);
474         int_threshold = nvnewa93;
475         hw_threshold = cellnum_93;
476
477     elseif counter == 6
478         %if nvnewa95 == 0
479             %disp('Alert--No peaks were detected for file')
```

```
480             %filename
481             %nv;
482             %widtha95 = 0;
483             %heighta95 = 0;
484             %absheighta95 = 0;
485             %ta95 = 0;
486             %dta95 = 0;
487             %indexa95 = 0;
488             %%datamatrixtotaltotal = 0;
489             %%timematrixtotaltotal = 0;
490             %signaldatamatrixtotaltotal = 0;
491             %signaltimematrixtotaltotal = 0;
492             %end
493
494       cellnum_95=0;
495       if nvnewa95 ~= 0
496         for k=1:84
497             %a=find(((k-1)*0.12 < heighta95) & (heighta95< k*0.12) & (heighta95-slope↙
    *widtha95-intercept>0.0));
498             a=find(((k-1)*0.12 <= heighta95) & (heighta95< k*0.12));
499             num=size(a);
500             histo_height_95(k,1)=num(1,1)*60/(endtime-starttime);
501             cellnum_95=cellnum_95+histo_height_95(k,1);
502         end
503
504         for k=1:200
505             %a=find(((k-1)*0.002 < widtha95) & (widtha95< k*0.002)& (heighta95-slope*↙
    widtha95-intercept>0.0));
506             a=find(((k-1)*0.002 <= widtha95) & (widtha95< k*0.002));
507             num=size(a);
508             histo_width_95(k,1)=num(1,1)*60/(endtime-starttime);
509         end
510
511         %ba95 = find(heighta95-slope*widtha95-intercept>0.0);
512         %finalpeaksheighta95 = heighta95(ba95(:));
513         %finalpeakswidtha95 = widtha95(ba95(:));
514         %finalpeakstimea95 = ta95(ba95(:));
515         %finalpeakstimeintervala95 = dta95(ba95(:));
516         %finalpeaksindexa95 = indexa95(ba95(:))
517         %finaltotalnumberofpeaksa95 = length(ba95);
518         %meanheighta95 = mean(finalpeaksheighta95);
519         %stdheighta95 = std(finalpeaksheighta95);
520
521         %[fidd,message] = fopen(heightfile,'r+','l');
522         %fseek(fidd,0,'bof');
523         %fwrite(fidd,heighta95,'float64');
524         %fclose(fidd);
525
526         filename1=strcat(root,'_95.txt');
```

```
527         filename2=strcat(root, '_pkcta95.txt');
528         filename3=strcat(root,'_histo_height_95.txt');
529         filename4=strcat(root,'_histo_width_95.txt');
530         %filename5=strcat(root,'_hwttii_95.txt');
531         filename6=strcat(root,'_mean_95.txt');
532         %filename7=strcat(root,'_rawdataplot_95.txt');
533         filename7=strcat(root,'_rawdataplot_95');
534         filename8=strcat(root,'_moundplot_95');
535         filename9=strcat(root,'_peakdataplot_95');
536         filename10=strcat(root,'_timedataplot_95');
537         filename11=strcat(root,'_peaktimeplot_95');
538         filename12=strcat(root,'_boundarytotal');
539         filename13=strcat(root,'_timemoundplot_95');
540         filename14=strcat(root,'_signalmoundplot_95');
541         filename15=strcat(root,'_signaltimemoundplot_95');
542         filename16=strcat(root,'_firstnumtotal');
543
544         eval(['dlmwrite(''' filename1 ''',[heighta95,widtha95,ta95,dta95,indexa95],''↙
    \t'')'])
545         nvnewa95;
546         cellnum_95;
547         eval(['dlmwrite(''' filename2 ''',[nvnewa95,cellnum_95,starttime,endtime],''\↙
    t'')'])
548         eval(['dlmwrite(''' filename3 ''',[x1,histo_height_95],''\t'')'])
549         eval(['dlmwrite(''' filename4 ''',[x2,histo_width_95],''\t'')'])
550         %eval(['dlmwrite(''' filename5 ''',[finalpeaksheighta95,finalpeakswidtha95,fi↙
    nalpeakstimea95,finalpeakstimeintervala95,finalpeaksindexa95],''\t'')'])
551         eval(['dlmwrite(''' filename6 ''',[meanv,stdv,boundary],''\t'')'])
552
553         %eval(['dlmwrite(''' filename7 ''',[data_smftotal,timetotal],''\t'')'])
554         fid = fopen(filename7,'w+','l');
555         switch A(i).Datatype;
556             case 4
557         fwrite(fid,data_smftotal,'float32');
558             case 5
559         fwrite(fid,data_smftotal,'float64');
560         end
561         %fwrite(fid,data_smftotal,'float64');
562         fclose(fid);
563
564         fiddddd = fopen(filename12,'w+','l');
565         switch A(i).Datatype;
566             case 4
567         fwrite(fiddddd,boundarytotal,'float32');
568             case 5
569         fwrite(fiddddd,boundarytotal,'float64');
570         end
571         %fwrite(fiddddd,boundarytotal,'float64');
572         fclose(fiddddd);
```

```
573
574         fidd = fopen(filename10,'w+','l');
575         fwrite(fidd,timetotal,'float32');
576         fclose(fidd);
577
578         ffiddddd = fopen(filename16,'w+','l');
579         switch A(i).Datatype;
580             case 4
581         fwrite(ffiddddd,firstnumtotal,'float32');
582             case 5
583         fwrite(ffiddddd,firstnumtotal,'float64');
584         end
585         %fwrite(ffiddddd,firstnumtotal,'float64');
586         fclose(ffiddddd);
587
588         %eval(['dlmwrite(''' filename8 ''',[datamatrixtotaltotal,timematrixtotaltotal↙
    ],''\t'')'])
589         gfid = fopen(filename8,'w+','l');
590         switch A(i).Datatype;
591             case 4
592         fwrite(gfid,datamatrixtotaltotal,'float32');
593             case 5
594         fwrite(gfid,datamatrixtotaltotal,'float64');
595         end
596         %fwrite(gfid,datamatrixtotaltotal,'float64');
597         fclose(gfid);
598
599         gfidd = fopen(filename13,'w+','l');
600         fwrite(gfidd,timematrixtotaltotal,'float32');
601         fclose(gfidd);
602
603         gfiddd = fopen(filename14,'w+','l');
604         switch A(i).Datatype;
605             case 4
606         fwrite(gfiddd,signaldatamatrixtotaltotal,'float32');
607             case 5
608         fwrite(gfiddd,signaldatamatrixtotaltotal,'float64');
609         end
610         %fwrite(gfiddd,signaldatamatrixtotaltotal,'float64');
611         fclose(gfiddd);
612
613         gfidddd= fopen(filename15,'w+','l');
614         fwrite(gfidddd,signaltimematrixtotaltotal,'float32');
615         fclose(gfidddd);
616
617         %eval(['dlmwrite(''' filename9 ''',[absheighta95,ta95],''\t'')'])
618         fiddd = fopen(filename9,'w+','l');
619         switch A(i).Datatype;
620             case 4
```

```
621          fwrite(fiddd,absheighta95,'float32');
622              case 5
623          fwrite(fiddd,absheighta95,'float64');
624          end
625          %fwrite(fiddd,absheighta95,'float64');
626          fclose(fiddd);
627
628          fidddd = fopen(filename11,'w+','l');
629          fwrite(fidddd,ta95,'float32');
630          fclose(fidddd);
631
632          datamatrixtotaltotal;
633
634          save_plot_variables(filename, heighta95, widtha95, x1, histo_height_95, x2, h↙
     isto_width_95);
635          int_threshold = nvnewa95;
636          hw_threshold = cellnum_95;
637
638       elseif nvnewa95 == 0
639          disp('Alert--No peaks were detected for file')
640          filename
641          int_threshold = nvnewa95;
642          hw_threshold = cellnum_95;
643        end %if nvnewa95 ~= 0
644
645      end %if counter == i
646      %fid = fopen('manualset.txt','a')
647      %fprintf(fid,'%18.14f %18.14f\n',widthh,heightt)
648      %fclose(fid)
649      %if counter == 1
650      %plot(widthhm,heighttm,'bd')
651      %grid on;
652      %axis([0 0.01 0 0.4])
653      %elseif counter ==2
654      %plot(widthha,heightta,'bd')
655      %grid on;
656      %axis([0 0.01 0 0.4])
657      %end
658  end %for counter = counter_vector(1):1:counter_vector(x(1,2))
659
660
```

```
1  %function [rms,meanv,stdv,n,nvm,indexm,tm,dtm,heightm,widthm,EOFM,diff,nevaluated,da↙
   tamatrixtotal,timematrixtotal,boundary,absheightm,signaldatamatrixtotal,signaltimema↙
   trixtotal]=analysis_newmod2(data,time,rmst,vt,w1,wt,stdv_th,totalpoints,nevaluated,n↙
   sm,counter,slope,intercept)
2  function [rms,meanv,stdv,n,nvs,indexs,ts,dts,heights,widths,EOFM,diff,nevaluated,dat↙
   amatrixtotal,timematrixtotal,boundary,firstnum,absheights,signaldatamatrixtotal,sign↙
   altimematrixtotal]=analysis_newmod54(data,time,rmst,vt,w1,wt,stdv_th,totalpoints,nev↙
   aluated,nsm,counter,slope,intercept)
3
4  %function for detecting and characterizing peaks in a trace.
5  %Data is an nx2 matrix, with the first column assumed to correspond to time
6  %rmst is the threshold voltage used to calculate the mean,std and rms of the noise i↙
   n the signal
7  %vt is a vector 1xv which contains the multiples of standard deviations of the noise↙
    intensity, which
8  %%%%will be used as an intensity threshold for counting peaks. The threshold is set ↙
   at
9  %%%% meanv+/-vt(i)*stdv, where vt=[2 3 5 10] for example.
10 %w1 is used to determine the size of the data point interval within which the progra↙
   m attempts to define
11 %%%the FWHM of the peak; it is also used when determining when two peaks are too clo↙
   se to be distinct
12 %wt is the the FWHM threshold for counting a particular peak, and it is in units of ↙
   seconds
13 %stdv_th is the number of noise intensity stds the peak intensity has to be in the c↙
   ase of two closely
14 %spaced peaks, in order to be considered for counting
15 %rms, meanv and stdv are the rms, mean and std of the noise, respectively
16 %nv is the number of cells above a given threshold. It is a vector equal to the size↙
    of vt, since
17 %%%you get a different nv for each one of the thresholds set by vt, i.e. its size is↙
    also 1xv
18 %index is an nv x v matrix. It contains the row number corresponding to each peak fo↙
   r each intensity
19 %%%threshold level set by vt
20 %t is an nv x v matrix containing the time of each peak detected for each one of the↙
    different thresholds
21 %dt is the time between a peak and its predecessor
22 %height is the intensity of the peak minus the mean background noise level
23 %width is the full width at half maximum of a given detected peak
24
25 %initialize variables
26 num=size(data);
27 totaln=num(1,1);
28 rms=0;
29 meanv=0;
30 stdv=0;
31 %data_sm=data;
32 data_sm(:,1) = time(:);
```

```
33  %size(time)
34  %size(data)
35  data_sm(:,2) = data(:);
36  j=0;
37
38  %%estimate noise rms, mean and std
39  if counter == 1
40      counter;
41      for i=1:totaln
42          if data_sm(i,2)<rmst
43              j=j+1;
44              rms=rms+(data_sm(i,2)*data_sm(i,2));
45              meanv=meanv+data_sm(i,2);
46          end
47      end
48      rms=sqrt(rms/j);
49      meanv=meanv/j;
50      for i=1:totaln
51          if data_sm(i,2)<rmst
52              stdv=stdv+(data_sm(i,2)-meanv)*(data_sm(i,2)-meanv);
53          end
54      end
55      stdv=sqrt(stdv/(j-1));
56
57  elseif counter == 2|3|4|5|6
58      counter;
59      if counter == 2
60          percent = 0.50;
61      elseif counter == 3
62          percent = 0.70;
63      elseif counter == 4
64          percent = 0.90;
65      elseif counter == 5
66          percent = 0.93;
67      elseif counter == 6
68          percent = 0.75;
69      end
70      sortedmatrix = sort(data_sm(:,2));
71      sortedmatrixsize = length(sortedmatrix);
72      absortedmatrixsize = round(percent*sortedmatrixsize);
73      firstnum = sortedmatrix(absortedmatrixsize);
74      comp = firstnum==sortedmatrix;
75      gg = find(comp);
76      absortedmatrixsize = max(gg);
77      meanv = mean(sortedmatrix(1:absortedmatrixsize));
78      stdv = std(sortedmatrix(1:absortedmatrixsize));
79      sqrddata = sortedmatrix .* sortedmatrix;
80      rms = sqrt(sum(sqrddata));
81  end
```

```
 82
 83  boundary = meanv + (vt*stdv);
 84  B = boundary >= data_sm(:,2);
 85  kk = find(B);
 86  s = size(kk);
 87
 88  if s(1,1) == 0
 89      n = 0;
 90  else
 91      n = max(kk);
 92  end
 93  if round(totalpoints) <= round(nevaluated + totaln)
 94      n = totaln;
 95      diff = 0;
 96      EOFM = 1;
 97  else
 98      nevaluated = nevaluated + n;
 99      EOFM = 0;
100      diff = totaln - n;
101  end
102
103  %boundary
104  %pause
105
106  if n ~= 0
107    abovethresholdindices = find(boundary < data_sm(1:n,2));
108    size(abovethresholdindices);
109    if length(abovethresholdindices) >= 1;
110        length(abovethresholdindices);
111        pointsabovethreshold = data_sm(abovethresholdindices(:),2);
112        %differenceinindices = diff(abovethresholdindices);
113        differenceinindices = abovethresholdindices(2:length(abovethresholdindices)) - ↙
     abovethresholdindices(1:length(abovethresholdindices)-1);
114        size(differenceinindices);
115        C = find(1 ~= differenceinindices);
116        numberofmounds = length(C) + 1;
117        signalmoundquantity = 0;
118        totalsignalnodecounter = 0;
119        for moundnumber = 1:1:numberofmounds;
120            if moundnumber == 1
121                start(moundnumber) = abovethresholdindices(1);
122                %timestart(moundnumber) = data_sm(abovethresholdindices(1),1);
123            else
124                start(moundnumber) = abovethresholdindices((C(moundnumber - 1))+1);
125                %timestart(moundnumber) = data_sm(abovethresholdindices((C(moundnumber↙
     - 1))+1),1);
126            end
127            if length(C) == 0
128                stop(moundnumber) = abovethresholdindices(length(abovethresholdindices↙
```

```
    ));
129             %timestop(moundnumber) = data_sm(abovethresholdindices(length(abovethr↙
    esholdindices)),1);
130         elseif moundnumber < numberofmounds
131             stop(moundnumber) = abovethresholdindices(C(moundnumber));
132             %timestop(moundnumber) = data_sm(abovethresholdindices(C(moundnumber))↙
    ,1);
133         elseif moundnumber == numberofmounds
134             stop(moundnumber) = abovethresholdindices(length(abovethresholdindices↙
    ));
135             %timestop(moundnumber) = data_sm(abovethresholdindices(length(abovethr↙
    esholdindices)),1);
136         end
137         moundnumber;
138         numberofmounds;
139         start;
140         stop;
141         datamatrix = data_sm(start(moundnumber):stop(moundnumber),2);
142         size(datamatrix);
143         timematrix = data_sm(start(moundnumber):stop(moundnumber),1);
144
145         if moundnumber == 1
146             datamatrixtotal = datamatrix(:);
147             timematrixtotal = timematrix(:);
148         elseif moundnumber > 1
149             datamatrixtotal(length(datamatrixtotal) + 1:length(datamatrixtotal) + ↙
    length(datamatrix)) = datamatrix(:);
150             timematrixtotal(length(timematrixtotal) + 1:length(timematrixtotal) + ↙
    length(timematrix)) = timematrix(:);
151             %pause
152         end
153
154         [absheightmound,indexmound] = max(datamatrix);
155         halfheight = (absheightmound + boundary)/2;
156
157         if length(datamatrix) == 1
158             fullwidthhalfmaximum = 0;
159
160         elseif length(datamatrix) >= 2
161                indexmindiffleft = indexmound;
162             while length(find(datamatrix(1:indexmindiffleft) >= halfheight)) > 0
163                 indexmindiffleft = indexmindiffleft - 1;
164                 if indexmindiffleft == 0
165                     indexmindiffleft = 1;
166                     break
167                 end
168              end
169             if halfheight > datamatrix(indexmindiffleft)
170             tmxl = timematrix(indexmindiffleft:indexmindiffleft+1);
```

```
171                 dmxl = datamatrix(indexmindiffleft:indexmindiffleft+1);
172                 halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'linear');
173                 elseif halfheight <= datamatrix(indexmindiffleft)
174                     if (start(moundnumber) == 1) & (length(datamatrix(1:indexmound)) >↙
      1)
175                         %tmxl = timematrix(1:indexmound);
176                         %dmxl = datamatrix(1:indexmound);
177                         dl = find(datamatrix(1) < datamatrix(1:indexmound));
178                         if length(dl) == 0
179                            halfheighttimeleft = timematrix(1)
180                         elseif length(dl) ~= 0
181                         dmxl(1) = datamatrix(1);
182                         dmxl(2) = datamatrix(dl(1));
183                         tmxl(1) = timematrix(1);
184                         tmxl(2) = timematrix(dl(1));
185                         halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'spline');
186                         end
187                     elseif start(moundnumber) > 1
188                         tmxl = data_sm(start(moundnumber)-1:start(moundnumber),1);
189                         dmxl = data_sm(start(moundnumber)-1:start(moundnumber),2);
190                         halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'linear');
191                     end
192                 end
193                 %halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'spline');
194
195                 indexmindiffright = indexmound;
196                  while length(find(datamatrix(indexmindiffright:length(datamatrix)) >=↙
      halfheight)) > 0
197                    indexmindiffright = indexmindiffright + 1;
198                    if indexmindiffright > length(timematrix)
199                       indexmindiffright = length(timematrix);
200                       break
201                    end
202                 end
203                 if halfheight > datamatrix(indexmindiffright)
204                 tmxr = timematrix(indexmindiffright-1:indexmindiffright);
205                 dmxr = datamatrix(indexmindiffright-1:indexmindiffright);
206                 halfheighttimeright = interp1(dmxr,tmxr,halfheight,'linear');
207                 elseif halfheight <= datamatrix(indexmindiffright)
208                     if (stop(moundnumber) == length(data_sm)) & (length(datamatrix(ind↙
    exmound:length(datamatrix))) > 1)
209                         %tmxr = timematrix(indexmound:length(timematrix));
210                         %dmxr = datamatrix(indexmound:length(datamatrix));
211                         dr = find(datamatrix(length(datamatrix)) < datamatrix(indexmou↙
    nd:length(datamatrix)));
212                         if length(dr) == 0
213                           halfheighttimeright = timematrix(length(datamatrix));
214                         elseif length(dr) ~= 0
215                         dmxr(1) = datamatrix(length(dr));
```

```
216                         dmxr(2) = datamatrix(length(datamatrix));
217                         tmxr(1) = timematrix(length(dr));
218                         tmxr(2) = timematrix(length(datamatrix));
219                         halfheighttimeright = interp1(dmxr,tmxr,halfheight,'spline');
220                         end
221                     elseif stop(moundnumber) < length(data_sm)
222                         tmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,1);
223                         dmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,2);
224                         halfheighttimeright = interp1(dmxr,tmxr,halfheight,'linear');
225                     end
226                 end
227                 %halfheighttimeright = interp1(dmxr,tmxr,halfheight,'spline');
228                 %halfheighttimeright = timematrix(indexmindiffright);
229                 if (start(moundnumber) == 1) & (length(datamatrix(1:indexmound)) == 1)
230                     fullwidthhalfmaximum = 2*(halfheighttimeright - timematrix(indexmo↙
    und));
231                 elseif (stop(moundnumber) == length(data_sm)) & (length(datamatrix(ind↙
    exmound:length(datamatrix))) == 1)
232                     fullwidthhalfmaximum = 2*(timematrix(indexmound) - halfheighttimel↙
    eft);
233                 else
234                     fullwidthhalfmaximum = halfheighttimeright - halfheighttimeleft;
235                 end
236             end %if length(datamatrix) == 1
237
238              slope,
239              intercept;
240
241                 %if moundnumber == 1
242                 %absheightmound
243                 %boundary
244                 %indexmound
245                 %indexmindiffleft
246                 %indexmindiffright
247                 %length(timematrix)
248                 %halfheight
249                 %halfheighttimeleft
250                 %halfheighttimeright
251                 %fullwidthhalfmaximum
252                 %absheightmound - boundary - (slope*fullwidthhalfmaximum) - intercept
253                 %pause
254                 %end
255
256             if (absheightmound - boundary - (slope*fullwidthhalfmaximum) - intercept >↙
     0) & (absheightmound - boundary - (stdv_th*stdv) > 0)
257             signalmoundquantity = signalmoundquantity + 1;
258
259         %-----------------------------------------------------------------------
260
```

```
261           datamatrixheightdifferences = datamatrix(2:length(datamatrix)) - datamatri↙
    x(1:length(datamatrix)-1);
262           signofdatamatrixheightdifferences = sign(datamatrixheightdifferences);
263           indicesofminusone = (find(-1 == signofdatamatrixheightdifferences)) + 1;
264           differenceofindicesofminusone = indicesofminusone(2:length(indicesofminuso↙
    ne)) - indicesofminusone(1:length(indicesofminusone)-1);
265           lastone = find(1 ~= differenceofindicesofminusone);
266           %
267           if length(indicesofminusone) == 0
268                 lastoneindices = length(datamatrix);
269           elseif length(indicesofminusone) > 0
270                 endoflastone = length(indicesofminusone);
271                 lastone;
272                 endoflastone;
273                 if length(lastone) == 0
274                     lastone = endoflastone;
275                 elseif length(lastone) > 0
276                     lastone = cat(1,lastone,endoflastone);
277                 end
278
279               %if length(differenceofindicesofminusone) == 0
280                 %lastoneindices = length(datamatrix);
281                %elseif length(differenceofindicesofminusone) > 0
282                 %if differenceofindicesofminusone(length(differenceofindicesofminus↙
    one)) == 1
283                 %endoflastone = length(differenceofindicesofminusone)+1;
284                 %lastone = cat(1,lastone,endoflastone);
285                 %end
286
287                 lastoneindices = indicesofminusone(lastone(:));
288                 beginningmatrixindice = [1];
289                 lastoneindices = cat(1,beginningmatrixindice,lastoneindices);
290           end
291
292           %if signalmoundquantity == 1
293               %timematrix
294               %boundary
295               %datamatrix
296               %signalmoundquantity
297               %absheightmound
298               %halfheighttimeleft
299               %indexmindiffleft
300               %halfheighttimeright
301               %indexmindiffright
302               %indexmound
303               %absheightmound - boundary - (slope*fullwidthhalfmaximum) - intercept
304               %lastoneindices
305               %pause
306               %end
```

```
307
308             signalnodecounter = 0;
309             signalnodeindice = 1;
310             b = 1;
311
312             for a = 1:1:length(lastoneindices)
313              if a == 1
314
315                 datatobetested = datamatrix(1:lastoneindices(a));
316                 timetobetested = timematrix(1:lastoneindices(a));
317                 [maxpoint,indexmaxpoint] = max(datatobetested);
318                 boundary1 = max(datatobetested(1),datatobetested(length(datatobetested↙
    )));
319                 halfheight = (maxpoint + boundary1)/2;
320
321               if length(datatobetested) == 1
322                 fullwidthhalfmaximum1 = 0;
323
324               elseif (datatobetested(:) == datamatrix(1:length(datatobetested))) & (i↙
    ndexmaxpoint == 1)
325                  halfheightm = (datatobetested(1) + datatobetested(length(datatobetest↙
    ed)))/2;
326                   indexmindiffright = indexmaxpoint;
327                  while datatobetested(indexmindiffright) >= halfheightm
328                    indexmindiffright = indexmindiffright + 1;
329                    if indexmindiffright > length(datatobetested)
330                       indexmindiffright = length(datatobetested);
331                       break
332                    end
333                  end
334                 if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
335                 halfheighttimeright = timetobetested(indexmindiffright);
336                 elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
    iffright)
337                 tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
338                 dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
339                 halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
340                 end
341                  if start(moundnumber) > 1
342                      tmxl = data_sm(start(moundnumber) - 1:start(moundnumber),1);
343                      dmxl = data_sm(start(moundnumber) - 1:start(moundnumber),2);
344                      halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
345                      fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
346                  elseif start(moundnumber) == 1
347                      fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1↙
    ));
348                  end
349                  boundary1 = datatobetested(length(datatobetested));
```

```
350
351             elseif (datatobetested(:) ~= datamatrix(1:length(datatobetested))) & (indexmaxpoint == 1)
352                 halfheightm = (datatobetested(1) + datatobetested(length(datatobetested)))/2;
353                 indexmindiffright = indexmaxpoint;
354                 while datatobetested(indexmindiffright) >= halfheightm
355                   indexmindiffright = indexmindiffright + 1;
356                   if indexmindiffright > length(datatobetested)
357                      indexmindiffright = length(datatobetested);
358                      break
359                   end
360               end
361                 if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffright)
362                 halfheighttimeright = timetobetested(indexmindiffright);
363             elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmindiffright)
364                 tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
365                 dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
366                 halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
367             end
368                 %fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1));
369                 fullwidthhalfmaximum1 = (halfheighttimeright - timetobetested(1));
370                 %boundary1 = datatobetested(length(datatobetested));
371                 boundary1 = maxpoint;
372
373               elseif (datatobetested(:) == datamatrix(length(datamatrix)-length(datatobetested)+1:length(datamatrix))) & (indexmaxpoint == length(datatobetested))
374                 halfheightm = (datatobetested(1) + datatobetested(length(datatobetested)))/2;
375                 indexmindiffleft = indexmaxpoint;
376                 while datatobetested(indexmindiffleft) >= halfheightm
377                     indexmindiffleft = indexmindiffleft - 1;
378                     if indexmindiffleft == 0
379                          indexmindiffleft = 1;
380                          break
381                      end
382                 end
383                 if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+1)
384                     halfheighttimeleft = timetobetested(indexmindiffleft);
385                 elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffleft+1)
386                  tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
387                  dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
388                  halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
389                 end
390                   if stop(moundnumber) < length(data_sm)
```

```
391                     tmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,1);
392                     dmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,2);
393                     halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
394                     fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
395                  elseif stop(moundnumber) == length(data_sm)
396                     fullwidthhalfmaximum1 = 2*(timetobetested(length(timetobetested)) ↙
    - halfheighttimeleft);
397                  end
398                  boundary1 = datatobetested(1);
399
400
401               elseif (length(datatobetested) >= 3) & (indexmaxpoint ~= 1) & (indexmax↙
    point ~= length(datatobetested))
402                   indexmindiffleft = indexmaxpoint;
403                  while datatobetested(indexmindiffleft) >= halfheight
404                      indexmindiffleft = indexmindiffleft - 1;
405                      if indexmindiffleft == 0
406                           indexmindiffleft = 1;
407                           break
408                       end
409                  end
410                  if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+↙
    1)
411                      halfheighttimeleft = timetobetested(indexmindiffleft);
412                  elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffl↙
    eft+1)
413                   tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
414                   dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
415                   halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'linear');
416                  end
417
418
419                       %if datatobetested(indexmindiffleft) == datatobetested(indexmindif↙
    fleft+1)
420                       %indexmaxpoint
421                       %indexmindiffleft
422                       %maxpoint
423                       %halfheight
424                       %datatobetested(indexmindiffleft)
425                       %datatobetested(indexmindiffleft+1)
426                       %datatobetested
427                       %timetobetested
428                       %lastoneindices
429                       %pause
430                       %end
431
432                  indexmindiffright = indexmaxpoint;
433                  while datatobetested(indexmindiffright) >= halfheight
434                    indexmindiffright = indexmindiffright + 1;
```

```
435                     if indexmindiffright > length(datatobetested)
436                        indexmindiffright = length(datatobetested);
437                        break
438                     end
439                  end
440                 if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
441                 halfheighttimeright = timetobetested(indexmindiffright);
442                 elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
    iffright)
443                 tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
444                 dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
445                 halfheighttimeright = interp1(dmxr,tmxr,halfheight,'linear');
446                 end
447               fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
448                 if (timetobetested(length(timetobetested)) == timematrix(length(timema↙
    trix))) & (datatobetested(1) < datatobetested(length(datatobetested)))
449                     boundary1 = datatobetested(1);
450                 end
451                  if (timetobetested(1) == timematrix(1)) & (datatobetested(1) > datato↙
    betested(length(datatobetested)))
452                     boundary1 = datatobetested(length(datatobetested));
453                  end
454           end %if length(datatobetested) == 1
455
456
457                 maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept;
458
459                 if (maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept >↙
     0) & (maxpoint - boundary1 - (stdv_th*stdv) > 0)
460                     signalnodecounter = signalnodecounter + 1;
461                     totalsignalnodecounter = totalsignalnodecounter + 1;
462                     signalnodeindice = lastoneindices(a);
463                     signalnode(signalnodecounter) = datamatrix(lastoneindices(a));
464                     absheights(totalsignalnodecounter) = maxpoint;
465                     heights(totalsignalnodecounter) = absheights(totalsignalnodecounte↙
    r) - boundary1;
466                     widths(totalsignalnodecounter) =  fullwidthhalfmaximum1;%timetobet↙
    ested(length(timetobetested)) - timetobetested(1);
467                     dts(totalsignalnodecounter) = 0;
468                     ts(totalsignalnodecounter) = timetobetested(indexmaxpoint);
469                     indexs(totalsignalnodecounter) = indexmaxpoint;
470                     nvs(totalsignalnodecounter) = length(heights);
471                     b = a;
472                 end %if maxpoint - (slope*fullwidthhalfmaximum1) - intercept > 0
473
474            elseif a > 1
475                a;
476                b;
```

```
477                 signalnodeindice;
478                 timematrix(signalnodeindice);
479                 minmatrix = find(datamatrix(lastoneindices(b:a-1)) < datamatrix(laston↙
    eindices(a)));
480
481                 if length(minmatrix) == 0
482                     bmax = b;
483
484                     for d = b:1:a-1
485                      for f = d+1:1:a
486
487                  if d >= bmax
488
489                  datatobetested = datamatrix(lastoneindices(d):lastoneindices(f));
490                  timetobetested = timematrix(lastoneindices(d):lastoneindices(f));
491                  [maxpoint,indexmaxpoint] = max(datatobetested);
492                  boundary1 = max(datatobetested(1),datatobetested(length(datatobeteste↙
    d)));
493                  halfheight = (maxpoint + boundary1)/2;
494
495              %if signalmoundquantity == 1
496                 %signalnodeindice
497                 %a
498                 %b
499                 %d
500                 %f
501                 %signalmoundquantity
502                 %absheightmound
503                 %halfheighttimeleft
504                 %indexmindiffleft
505                 %halfheighttimeright
506                 %indexmindiffright
507                 %maxpoint
508                 %indexmound
509                 %absheightmound - boundary - (slope*fullwidthhalfmaximum) - intercept
510                 %lastoneindices
511                 %pause
512                 %end
513
514
515              if length(datatobetested) == 1
516                 fullwidthhalfmaximum1 = 0;
517
518                elseif (datatobetested(:) == datamatrix(1:length(datatobetested))) & (i↙
    ndexmaxpoint == 1);
519                  halfheightm = (datatobetested(1) + datatobetested(length(datatobetest↙
    ed)))/2;
520                   indexmindiffright = indexmaxpoint;
521                  while datatobetested(indexmindiffright) >= halfheightm
```

```
522                     indexmindiffright = indexmindiffright + 1;
523                     if indexmindiffright > length(datatobetested)
524                        indexmindiffright = length(datatobetested);
525                        break
526                     end
527                   end
528                  if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
     ight)
529                  halfheighttimeright = timetobetested(indexmindiffright);
530                  elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
     iffright)
531                  tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
532                  dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
533                  halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
534                  end
535                   if start(moundnumber) > 1
536                       tmxl = data_sm(start(moundnumber) - 1:start(moundnumber),1);
537                       dmxl = data_sm(start(moundnumber) - 1:start(moundnumber),2);
538                       halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
539                       fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
540                   elseif start(moundnumber) == 1
541                       fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1↙
     ));
542                   end
543                   boundary1 = datatobetested(length(datatobetested));
544
545              elseif (datatobetested(:) ~= datamatrix(1:length(datatobetested))) & (ind↙
     exmaxpoint == 1);
546                   halfheightm = (datatobetested(1) + datatobetested(length(datatobetest↙
     ed)))/2;
547                    indexmindiffright = indexmaxpoint;
548                   while datatobetested(indexmindiffright) >= halfheightm
549                     indexmindiffright = indexmindiffright + 1;
550                     if indexmindiffright > length(datatobetested)
551                        indexmindiffright = length(datatobetested);
552                        break
553                    end
554                  end
555                  if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
     ight)
556                  halfheighttimeright = timetobetested(indexmindiffright);
557                  elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
     iffright)
558                  tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
559                  dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
560                  halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
561                  end
562                  %fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1));
563                  fullwidthhalfmaximum1 = (halfheighttimeright - timetobetested(1));
```

```
564                 %boundary1 = datatobetested(length(datatobetested));
565                 boundary1 = maxpoint;
566
567
568               elseif (datatobetested(:) == datamatrix(length(datamatrix)-length(datat↙
    obetested)+1:length(datamatrix))) & (indexmaxpoint == length(datatobetested))
569                 halfheightm = (datatobetested(1) + datatobetested(length(datatobeteste↙
    d)))/2;
570                 indexmindiffleft = indexmaxpoint;
571               while datatobetested(indexmindiffleft) >= halfheightm
572                   indexmindiffleft = indexmindiffleft - 1;
573                   if indexmindiffleft == 0
574                       indexmindiffleft = 1;
575                       break
576                    end
577               end
578               if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+↙
    1)
579                   halfheighttimeleft = timetobetested(indexmindiffleft);
580               elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffl↙
    eft+1)
581                 tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
582                 dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
583                 halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
584               end
585                 if stop(moundnumber) < length(data_sm)
586                   tmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,1);
587                   dmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,2);
588                   halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
589                   fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
590                 elseif stop(moundnumber) == length(data_sm)
591                   fullwidthhalfmaximum1 = 2*(timetobetested(length(timetobetested)) ↙
    - halfheighttimeleft);
592                 end
593                 boundary1 = datatobetested(1);
594
595
596               elseif (length(datatobetested) >= 3) & (indexmaxpoint ~= 1) & (indexmax↙
    point ~= length(datatobetested))
597                 indexmindiffleft = indexmaxpoint;
598               while datatobetested(indexmindiffleft) >= halfheight
599                   indexmindiffleft = indexmindiffleft - 1;
600                   if indexmindiffleft == 0
601                       indexmindiffleft = 1;
602                       break
603                    end
604               end
605               if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+↙
    1)
```

```
606                     halfheighttimeleft = timetobetested(indexmindiffleft);
607                 elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffl↙
    eft+1)
608                  tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
609                  dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
610                  halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'linear');
611                 end
612
613
614                      %if datatobetested(indexmindiffleft) == datatobetested(indexmindif↙
    fleft+1)
615                      %indexmaxpoint
616                      %indexmindiffleft
617                      %maxpoint
618                      %halfheight
619                      %datatobetested(indexmindiffleft)
620                      %datatobetested(indexmindiffleft+1)
621                      %datatobetested
622                      %timetobetested
623                      %lastoneindices
624                      %pause
625                      %end
626
627                   indexmindiffright = indexmaxpoint;
628                   while datatobetested(indexmindiffright) >= halfheight
629                     indexmindiffright = indexmindiffright + 1;
630                     if indexmindiffright > length(datatobetested)
631                        indexmindiffright = length(datatobetested);
632                        break
633                     end
634                   end
635                  if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
636                  halfheighttimeright = timetobetested(indexmindiffright);
637                  elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
    iffright)
638                  tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
639                  dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
640                  halfheighttimeright = interp1(dmxr,tmxr,halfheight,'linear');
641                  end
642                fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
643                  if (timetobetested(length(timetobetested)) == timematrix(length(timema↙
    trix))) & (datatobetested(1) < datatobetested(length(datatobetested)))
644                      boundary1 = datatobetested(1);
645                  end
646                   if (timetobetested(1) == timematrix(1)) & (datatobetested(1) > datato↙
    betested(length(datatobetested)))
647                      boundary1 = datatobetested(length(datatobetested));
648                   end
```

```
649           end %if length(datatobetested) == 1
650
651                %if signalmoundquantity == 1
652                %signalmoundquantity
653                %a
654                %b
655                %d
656                %f
657                %indexmaxpoint
658                %maxpoint
659                %indexmindiffleft
660                %halfheighttimeleft
661                %indexmindiffright
662                %halfheighttimeright
663                %fullwidthhalfmaximum1
664                %maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept
665                %lastoneindices
666                %pause
667                %end
668
669              maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept;
670
671              if (maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept > ↙
     0) & (maxpoint - boundary1 - (stdv_th*stdv) > 0)
672                   signalnodecounter = signalnodecounter + 1;
673                   totalsignalnodecounter = totalsignalnodecounter + 1;
674                   %signalnodeindice = lastoneindices(a);
675                   %signalnode(signalnodecounter) = datamatrix(lastoneindices(a));
676                   signalnodeindice = lastoneindices(f);
677                   signalnode(signalnodecounter) = datamatrix(lastoneindices(f));
678                   absheights(totalsignalnodecounter) = maxpoint;
679                   heights(totalsignalnodecounter) = absheights(totalsignalnodecounte↙
     r) - boundary1;
680                   widths(totalsignalnodecounter) = fullwidthhalfmaximum1;%timetobete↙
     sted(length(timetobetested)) - timetobetested(1);
681                   dts(totalsignalnodecounter) = 0;
682                   ts(totalsignalnodecounter) = timetobetested(indexmaxpoint);
683                   indexs(totalsignalnodecounter) = indexmaxpoint;
684                   nvs(totalsignalnodecounter) = length(heights);
685                   bb = f;
686                        %if (signalmoundquantity == 1)
687                        %%signalmoundquantity
688                        %maxpoint
689                        %boundary1
690                        %indexmaxpoint
691                        %halfheight
692                        %halfheighttimeleft
693                        %halfheighttimeright
694                        %fullwidthhalfmaximum1
```

```
695                         %maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - inter↙
    cept
696                         %ts
697                         %timetobetested(1)
698                         %timetobetested(length(timetobetested))
699                         %d
700                         %f
701                         %last
702                         %datamatrix(lastoneindices(d:f))
703                         %datamatrixofminimums
704                         %pause
705                         %end
706                     if bmax < bb
707                         bmax = bb;
708                     end
709                 end %if maxpoint - (slope*fullwidthhalfmaximum1) - intercept > 0
710
711             end  %if d >= bmax
712           end %for f = d+1:1:a
713       end %for d = b:1:a-1
714       b = bmax;
715                 elseif length(minmatrix) > 0  %if length(minmatrix) == 0
716                     minmatrix1 = find(datamatrix(lastoneindices(1:a-1)) < datamatrix(l↙
    astoneindices(a)));
717                     leftindexofdatatobetested = max(signalnodeindice,lastoneindices(mi↙
    nmatrix1(length(minmatrix1))));
718
719             %if signalmoundquantity == 4
720                 %minmatrix1 = find(datamatrix(lastoneindices(1:a-1)) < datamatrix(last↙
    oneindices(a)))
721                 %leftindexofdatatobetested = max(signalnodeindice,lastoneindices(minma↙
    trix1(length(minmatrix1))))
722                 %signalnodeindice
723                 %a
724                 %b
725                 %signalmoundquantity
726                 %absheightmound
727                 %halfheighttimeleft
728                 %indexmindiffleft
729                 %halfheighttimeright
730                 %indexmindiffright
731                 %indexmound
732                 %absheightmound - boundary - (slope*fullwidthhalfmaximum) - intercept
733                 %lastoneindices
734                 %pause
735                 %end
736
737                 datatobetested = datamatrix(leftindexofdatatobetested:lastoneindices(a↙
    ));
```

```
738                 timetobetested = timematrix(leftindexofdatatobetested:lastoneindices(a↙
    ));
739                 [maxpoint,indexmaxpoint] = max(datatobetested);
740                 boundary1 = max(datatobetested(1),datatobetested(length(datatobetested↙
    )));
741                 halfheight = (maxpoint + boundary1)/2;
742
743               if length(datatobetested) == 1
744                 fullwidthhalfmaximum1 = 0;
745
746               elseif (datatobetested(:) == datamatrix(1:length(datatobetested))) & (i↙
    ndexmaxpoint == 1)
747                  halfheightm = (datatobetested(1) + datatobetested(length(datatobetest↙
    ed)))/2;
748                   indexmindiffright = indexmaxpoint;
749                  while datatobetested(indexmindiffright) >= halfheightm
750                    indexmindiffright = indexmindiffright + 1;
751                    if indexmindiffright > length(datatobetested)
752                       indexmindiffright = length(datatobetested);
753                       break
754                    end
755                  end
756                 if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
757                 halfheighttimeright = timetobetested(indexmindiffright);
758                 elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
    iffright)
759                 tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
760                 dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
761                 halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
762                 end
763                  if start(moundnumber) > 1
764                      tmxl = data_sm(start(moundnumber) - 1:start(moundnumber),1);
765                      dmxl = data_sm(start(moundnumber) - 1:start(moundnumber),2);
766                      halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
767                      fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
768                  elseif start(moundnumber) == 1
769                      fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1↙
    ));
770                  end
771                  boundary1 = datatobetested(length(datatobetested));
772
773             elseif (datatobetested(:) ~= datamatrix(1:length(datatobetested))) & (ind↙
    exmaxpoint == 1)
774                  halfheightm = (datatobetested(1) + datatobetested(length(datatobetest↙
    ed)))/2;
775                   indexmindiffright = indexmaxpoint;
776                  while datatobetested(indexmindiffright) >= halfheightm
777                    indexmindiffright = indexmindiffright + 1;
```

```
778                     if indexmindiffright > length(datatobetested)
779                        indexmindiffright = length(datatobetested);
780                        break
781                     end
782                 end
783                   if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
784                   halfheighttimeright = timetobetested(indexmindiffright);
785               elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmindiffr↙
    ight)
786                   tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
787                   dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
788                   halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
789               end
790                   %fullwidthhalfmaximum1 = 2*(halfheighttimeright - timetobetested(1));
791                   fullwidthhalfmaximum1 = (halfheighttimeright - timetobetested(1));
792                   %boundary1 = datatobetested(length(datatobetested));
793                   boundary1 = maxpoint;
794
795
796                elseif (datatobetested(:) == datamatrix(length(datamatrix)-length(datat↙
    obetested)+1:length(datamatrix))) & (indexmaxpoint == length(datatobetested))
797                 halfheightm = (datatobetested(1) + datatobetested(length(datatobeteste↙
    d)))/2;
798                 indexmindiffleft = indexmaxpoint;
799                 while datatobetested(indexmindiffleft) >= halfheightm
800                     indexmindiffleft = indexmindiffleft - 1;
801                     if indexmindiffleft == 0
802                          indexmindiffleft = 1;
803                          break
804                      end
805                 end
806                 if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+↙
    1)
807                     halfheighttimeleft = timetobetested(indexmindiffleft);
808                 elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffl↙
    eft+1)
809                  tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
810                  dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
811                  halfheighttimeleft = interp1(dmxl,tmxl,halfheightm,'linear');
812                 end
813                   if stop(moundnumber) < length(data_sm)
814                      tmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,1);
815                      dmxr = data_sm(stop(moundnumber):stop(moundnumber)+1,2);
816                      halfheighttimeright = interp1(dmxr,tmxr,halfheightm,'linear');
817                      fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
818                   elseif stop(moundnumber) == length(data_sm)
819                      fullwidthhalfmaximum1 = 2*(timetobetested(length(timetobetested)) ↙
    - halfheighttimeleft);
```

```
820                 end
821                 boundary1 = datatobetested(1);
822
823
824               elseif (length(datatobetested) >= 3) & (indexmaxpoint ~= 1) & (indexmax↙
    point ~= length(datatobetested))
825                 indexmindiffleft = indexmaxpoint;
826                 while datatobetested(indexmindiffleft) >= halfheight
827                     indexmindiffleft = indexmindiffleft - 1;
828                     if indexmindiffleft == 0
829                          indexmindiffleft = 1;
830                          break
831                      end
832                 end
833                 if datatobetested(indexmindiffleft) == datatobetested(indexmindiffleft+↙
    1)
834                     halfheighttimeleft = timetobetested(indexmindiffleft);
835                 elseif datatobetested(indexmindiffleft) ~= datatobetested(indexmindiffl↙
    eft+1)
836                  tmxl = timetobetested(indexmindiffleft:indexmindiffleft+1);
837                  dmxl = datatobetested(indexmindiffleft:indexmindiffleft+1);
838                  halfheighttimeleft = interp1(dmxl,tmxl,halfheight,'linear');
839                 end
840
841
842                      %if datatobetested(indexmindiffleft) ~= datatobetested(indexmindif↙
    fleft+1)
843                      %indexmaxpoint
844                      %indexmindiffleft
845                      %maxpoint
846                      %halfheight
847                      %datatobetested(indexmindiffleft)
848                      %datatobetested(indexmindiffleft+1)
849                      %datatobetested
850                      %timetobetested
851                      %lastoneindices
852                      %pause
853                      %end
854
855                   indexmindiffright = indexmaxpoint;
856                   while datatobetested(indexmindiffright) >= halfheight
857                     indexmindiffright = indexmindiffright + 1;
858                     if indexmindiffright > length(datatobetested)
859                        indexmindiffright = length(datatobetested);
860                        break
861                     end
862                   end
863                  if datatobetested(indexmindiffright-1) == datatobetested(indexmindiffr↙
    ight)
```

```
864                 halfheighttimeright = timetobetested(indexmindiffright);
865                 elseif datatobetested(indexmindiffright-1) ~= datatobetested(indexmind↙
    iffright)
866                 tmxr = timetobetested(indexmindiffright-1:indexmindiffright);
867                 dmxr = datatobetested(indexmindiffright-1:indexmindiffright);
868                 halfheighttimeright = interp1(dmxr,tmxr,halfheight,'linear');
869                 end
870               fullwidthhalfmaximum1 = halfheighttimeright - halfheighttimeleft;
871                 if (timetobetested(length(timetobetested)) == timematrix(length(timema↙
    trix))) & (datatobetested(1) < datatobetested(length(datatobetested)))
872                     boundary1 = datatobetested(1);
873                 end
874                  if (timetobetested(1) == timematrix(1)) & (datatobetested(1) > datato↙
    betested(length(datatobetested)))
875                     boundary1 = datatobetested(length(datatobetested));
876                  end
877           end %if length(datatobetested) == 1
878
879
880           %if signalmoundquantity == 1
881                %signalmoundquantity
882                %a
883                %b
884                %d
885                %f
886                %indexmaxpoint
887                %maxpoint
888                %indexmindiffleft
889                %halfheighttimeleft
890                %indexmindiffright
891                %halfheighttimeright
892                %fullwidthhalfmaximum1
893                %maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept
894                %lastoneindices
895                %pause
896                %end
897
898
899               if (maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept > ↙
    0) & (maxpoint - boundary1 - (stdv_th*stdv) > 0)
900                     signalnodecounter = signalnodecounter + 1;
901                     totalsignalnodecounter = totalsignalnodecounter + 1;
902                     signalnodeindice = lastoneindices(a);
903                     signalnode(signalnodecounter) = datamatrix(lastoneindices(a));
904                     absheights(totalsignalnodecounter) = maxpoint;
905                     heights(totalsignalnodecounter) = absheights(totalsignalnodecounte↙
    r) - boundary1;
906                     widths(totalsignalnodecounter) = fullwidthhalfmaximum1;%timetobete↙
    sted(length(timetobetested)) - timetobetested(1);
```

```
907                     dts(totalsignalnodecounter) = 0;
908                     ts(totalsignalnodecounter) = timetobetested(indexmaxpoint);
909                     indexs(totalsignalnodecounter) = indexmaxpoint;
910                     nvs(totalsignalnodecounter) = length(heights);
911
912                         %if signalmoundquantity == 1
913                         %signalmoundquantity
914                         %maxpoint
915                         %boundary1
916                         %indexmindiffleft
917                         %halfheighttimeleft
918                         %indexmindiffright
919                         %halfheighttimeright
920                         %fullwidthhalfmaximum1
921                         %maxpoint - boundary1 - (slope*fullwidthhalfmaximum1) - intercept
922                         %ts
923                         %pause
924                         %end
925                     b = a;
926                 end %if maxpoint - (slope*fullwidthhalfmaximum1) - intercept > 0
927               end %if length(minmatrix) == 0
928             end %if a == 1
929         end %for a = 1:1:length(lastoneindices)
930
931
932     %-------------------------------------------------------------------------------------------
933               if signalnodecounter == 0
934                 totalsignalnodecounter = totalsignalnodecounter + 1;
935                 signalnodeindice = lastoneindices(a);
936                 [maxpoint indexmaxpoint] = max(datamatrix);
937                 absheights(totalsignalnodecounter) = maxpoint;
938                 heights(totalsignalnodecounter) = absheights(totalsignalnodecounter) - boundary;
939                 widths(totalsignalnodecounter) = fullwidthhalfmaximum; %timematrix(length(timetobetested)) - timematrix(1);
940                 dts(totalsignalnodecounter) = 0;
941                 ts(totalsignalnodecounter) = timematrix(indexmaxpoint);
942                 indexs(totalsignalnodecounter) = indexmaxpoint;
943                 nvs(totalsignalnodecounter) = length(heights);
944               end
945
946
947
948             absheightm(signalmoundquantity) = absheightmound;
949             heightm(signalmoundquantity) = absheightmound - boundary;
950             widthm(signalmoundquantity) = timematrix(length(timematrix)) - timematrix(1);
```

```
951             dtm(signalmoundquantity) = 0;
952             tm(signalmoundquantity) = timematrix(indexmound);
953             indexm(signalmoundquantity) = indexmound;
954             nvm(signalmoundquantity) = length(heightm);
955
956             if signalmoundquantity == 1
957                 signaldatamatrixtotal = datamatrix(:);
958                 signaltimematrixtotal = timematrix(:);
959             else
960                 signaldatamatrixtotal(length(signaldatamatrixtotal) + 1:length(signald↙
    atamatrixtotal) + length(datamatrix)) = datamatrix(:);
961                 signaltimematrixtotal(length(signaltimematrixtotal) + 1:length(signalt↙
    imematrixtotal) + length(timematrix)) = timematrix(:);
962             end
963           end %if absheightmound - (slope*fullwidthhalfmaximum) - intercept > 0
964
965       end %for moundnumber = 1:1:numberofmounds;
966
967       totalsignalnodecounter;
968       if totalsignalnodecounter > 0
969           absheights = absheights';
970           heights = heights';
971           widths = widths';
972           dts = dts';
973           ts = ts';
974           nvs = length(heights');
975           indexs = indexs';
976       elseif totalsignalnodecounter == 0
977           absheights = 0;
978           heights = 0;
979           widths = 0;
980           dts = 0;
981           ts = 0;
982           nvs = 0;
983           indexs = 0;
984       end
985       signalmoundquantity;
986       if signalmoundquantity >= 1
987           absheightm = absheightm(:);
988           heightm = heightm(:);
989           widthm = widthm(:);
990           dtm = dtm(:);
991           tm = tm(:);
992           nvm = length(heightm(:));
993           indexm = indexm(:);
994       else
995           absheightm = 0;
996           heightm = 0;
997           widthm = 0;
```

```
 998            dtm = 0;
 999            tm = 0;
1000            indexm = 0;
1001            nvm = 0;
1002            %datamatrixtotal = 0;
1003            %timematrixtotal = 0;
1004            signaldatamatrixtotal = 0;
1005            signaltimematrixtotal = 0;
1006         end %if signalmoundquantity >= 1
1007           else
1008            absheights = 0;
1009            heights = 0;
1010            widths = 0;
1011            dts = 0;
1012            ts = 0;
1013            nvs = 0;
1014            indexs = 0;
1015            absheightm = 0;
1016            heightm = 0;
1017            widthm = 0;
1018            dtm= 0;
1019            tm = 0;
1020            indexm = 0;
1021            nvm = 0;
1022            datamatrixtotal = 0;
1023            timematrixtotal = 0;
1024            signaldatamatrixtotal = 0;
1025            signaltimematrixtotal = 0;
1026    end %if length(abovethresholdindices) >= 1;
1027      else
1028            absheights = 0;
1029            heights = 0;
1030            widths = 0;
1031            dts = 0;
1032            ts = 0;
1033            nvs = 0;
1034            indexs = 0;
1035            absheightm = 0;
1036            heightm = 0;
1037            widthm = 0;
1038            dtm = 0;
1039            tm = 0;
1040            indexm = 0;
1041            nvm = 0;
1042            datamatrixtotal = 0;
1043            timematrixtotal = 0;
1044            signaldatamatrixtotal = 0;
1045            signaltimematrixtotal = 0;
1046  end %if n -= 0
```

```
%datamatrixtotal
%timematrixtotal
%boundary
```

Thesis Proposal

Development of an In Vivo Flow Cytometer and Application to Investigating Tumor Metastasis

by

John Novak

Abstract and Specific Aims

This proposal discusses the design and development of a new device called an in vivo flow cytometer, the determination of its system performance characteristics, and the application of this device to address several scientific questions concerning malignant tumor metastasis. The in vivo flow cytometer will be comprised of a scanning laser confocal microscope to locate subcutaneous blood vessels from which data will be acquired, and a stationary probe beam to detect and count flowing cells. The data that will be obtained includes number of circulating tumor cells in the circulatory system of a mammal (mouse) with a malignant turmor, and flow velocity range of the tumor cells in the blood flow. This flow cytometer will differ from existing cytometers in that the device will acquire data on the tumor cells in the circulatory system without extracting them from the body. Interfaced with the in vivo flow cytometer will be a noninvasive temperature probe. This device will determine the temperature of the soft tissue within the probe volume by examining the Raman scattering signal from the water in the soft tissue. The performance of this noninvasive probe will be verified against theoretical and numerical models. The detection circuitry of the cytometer will be customed designed and built in the lab. This will be necessary because of the probable signal variability resulting from light diffraction affects of the blood and soft tissue and the variability in flow velocity and radial location in the blood vessel of the circulating tumor cells. Once constructed, system performance characteristics will be determined via six system characterization tests. The six system characterization tests will establish various limits of in vivo detectability of fluorescent microspheres and fluorescently labeled tumor cells by the device. The fluorescent labeling of the tumor cells will be accomplished by injecting Cy5.5-conjugated PEQ226.5 antibodies through the tail vein of the mouse. Further refinement of the

device will take place if the system performance tests do not yield satisfactory results. Once the device is functioning properly, data acquisition will be performed. The data from the cytometer, in conjunction with tumor size and lymph node data, will be analyzed to determine if there is correlation between circulating tumor cell count and various aspects of malignant tumor metastasis. For example, the flow velocity range data will be used to try to determine if tumor cells travel through the blood stream similar to the way leukocytes do. If this is true, then the mechanisms which contribute to binding leukocytes to the vascular wall to fight infection might also contribute to tumor metastasis.

Thus, in summary, the specific aims of the project are:

1) Development of an in vivo flow cytometer comprised of a scanning laser confocal microscope to locate an appropriate blood vessel (in a mouse) from which circulating tumor cell count and tumor cell flow velocity data can be acquired, and a flow cytometer which will acquire the data.
2) Development of a noninvasive temperature probe which utilizes Raman scattering by the water molecules in the soft tissue to determine soft tissue temperature, and verification of performance via analytical and numerical modeling.
3) Development of detection circuitry for the flow cytometer.
4) System characterization of the completed device and data acquisition. Data will include circulating tumor cell count and tumor cell flow profile obtained by the cytometer, tumor size and lymph node data obtained manually.
5) Analysis of the data to determine if there is a correlation between circulating tumor cell count and various aspects of malignant tumor metastasis (for mice).

Thesis Committee:

Dr. Charles Lin (Thesis Advisor)
Principal Research Scientist, Wellman Labs (MGH)

Prof C. Forbes Dewey (Committee Chair)
Professor of Mechanical and Bioengineering Engineering, MIT

Prof Roger Kamm
Professor of Mechanical and Bioengineering Engineering, MIT

Significance

Despite the advances made in malignant tumor (i.e. cancer) research over the last several decades, many questions still remain to be answered about this medical malady. For example, it is hypothesized that the shedding of tumor cells into the circulatory system is one of the key steps in cancer metastasis, but it is not known at what stage or stages in tumor growth that this occurs [Brandt BH, Schmidt H, de Angelis G, Zanker KS. Predictive laboratory diagnostics in oncology utilizing blood-borne cancer cells ± current best practice and unmet needs. Cancer Letters 162, S11-16 (2001)]. It is also not known if the circulating tumor cell (CTC) count is representative of metastatic potential of the tumor, and if the CTC count is indicative of tumor burden [ibid]. In addition, it remains to be discovered what correlation exists between CTC count and the patient's (human as well as animal) response to malignant tumor therapy (such as hyperthermia, ultrasound, and photodynamic treatment).

One of the current methods for the detection and counting of circulating tumor cells is (ex vivo) flow cytometry. This method involves the extraction of blood cells from the patient, the fluorescent labeling of specific cell populations, and the insertion of the blood cells into an flow cytometer. The standard flow cytometer, comprised of a light source (usually a laser), optics, light filters, light detectors, fluid lines, and electronics, passes the cells of the withdrawn blood sample in a single file through the light source and determines, via analysis of the fluorescent signal and the forward and orthogonally scattered light, the types and number of cells present. Although this method has its merits, it only provides a single time sample, with a significant time delay between blood withdrawl and analysis results. In addition, there is ample opportunity for sample contamination, as well as damage to the cells of interest, resulting in erroneous data [Personal communication with MIT flow cytometry lab]. Another contemporary method employed to detect and, hence, count tumor cells is reverse transcriptase-polymerase chain reaction (RT-PCR). This method is similar to ex vivo flow cytometry in that a blood sample is extracted from the patient (or animal specimen) and analysis of the blood is performed ex vivo. RT-PCR involves copying RNA to DNA and then amplifying the DNA. Cancer cells are detected and counted via the disparity of DNA structure between cancer cells and normal cells (i.e. cancer cells have different PCR products). This method has a detection sensitivity for LNCaP cells (the cancer cells that will be used in this study) of 1 in 10^6 [Ghossein RA, Bhattacharya S. Molecular detection and characterisation of

circulating tumour cells and micrometastases in solid tumours. European Journal of Cancer 36, 1681-1694 (2000).] However, like ex vivo cytometry, this technique affords the opportunity for sample contamination, and provides only a single time sample, with a significant time delay between blood withdrawl and analysis results.

To remedy these problems, we propose to develop a new method for the detection and counting of circulating tumor cells by in vivo flow cytometry, where analysis of the tumor cells would be performed with the tumor cells remaining in the circulatory system of the body. This method would allow for continuous acquisition of data with no time delay, as well as no possibility for the contamination of the sample. Although this method would allow data acquisition on the cells of interest only through fluorescent tagging (since forward and orthogonally scattered light will not be able to be collected), this should not prove to be too heavy of a penalty, since the information desired can be obtained by analyzing the fluorescent signals recorded. In short, this method will allow for acquisition of data (from mice) which could be used to answer or help to answer the above questions concerning time course of tumor cell shedding, metastatic potential of the tumor, tumor burden, and degree of correlation of CTC count with animal responsivity to tumor therapy. If successful, this process could then be applied to human subjects in the future, although it is hoped that the data obtained from this present study can be applied to humans as well.

Approach

Optical platform The optical layout of the proposed instrument is shown in Figure 1. Shown in black are the components of a scanning laser confocal microscope which will be used to visualize and identify blood vessels from which the data will be acquired, and shown in red are the components of the flow cytometer which will identify and count the fluorescently labeled cells. Imaging vasculature via a confocal microscope is possible due to intrinsic contrast mechanisms such as back scattering and absorption contrast (in combination with cross polarization imaging). It is also possible to inject a fluorescent vascular contrast agent such as ICG. Laser 1 is the source for the scanning beam of visualization. The excitation wavelength of the scanning beam will be 780 nm (diode laser) if ICG is used as the vascular marker, and anywhere from 780 nm to 1064 nm (Nd:YAG laser) if imaging by backscattering. Laser 2 is the source of the stationary beam of the flow cytometer. The power output of

Laser 1 and Laser 2 will be determined by the maximum steady state temperature of the soft tissue being irradiated by the lasers. (A preliminary thermal analysis indicates a maximum

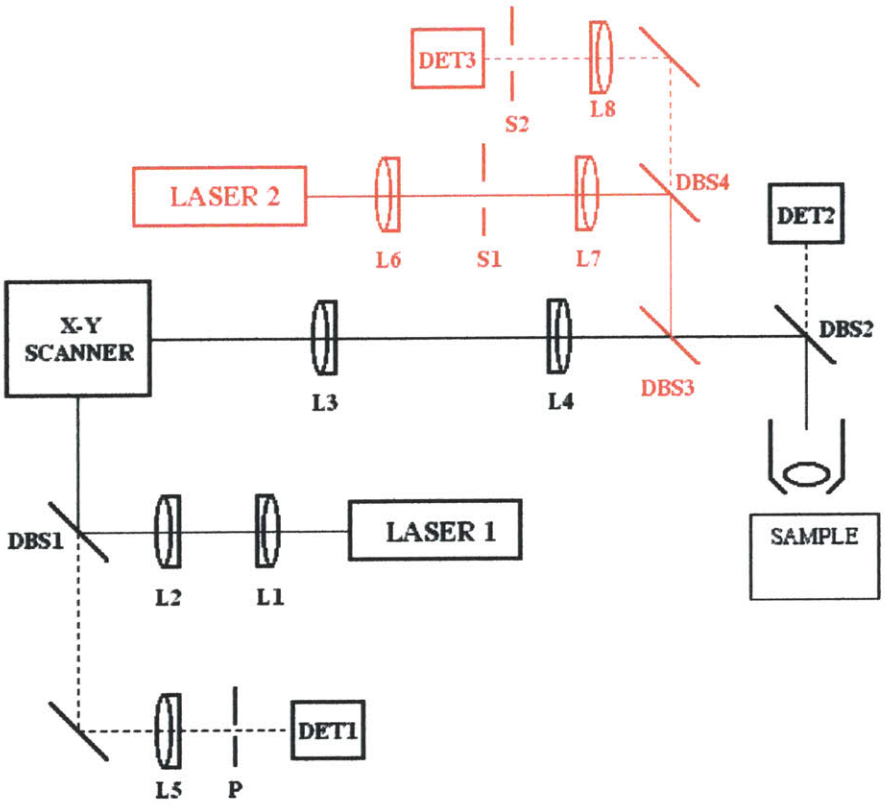

Fig. 1. Optical layout of the proposed instrument. Shown in black are components of a scanning laser confocal microscope used to visualize and identify blood vessels. Shown in red are components of the stationary beam used for flow cytometry measurement. L1-L8: lenses. (L6 is a cylindrical lens.) DBS1-DBS4: dichroic beam splitters. P: confocal pinhole aperture. S1-S2: confocal slit apertures. DET1-DET3: photodetectors.

steady state temperature below 40^0 C for the soft tissue for a combined 20 mW energy deposition rate by Laser 1 and Laser 2). Either a HeNe laser (633 nm) or a diode laser (660-690 nm) will be used to excite the fluorescently tagged cells of interest. The fluorescent tags will be Cy5.5-conjugated antibodies for experiments requiring the labeling of only one cell population. Cy5.5 is the probe of choice because its long absorption and emission spectrum will allow for transmission through the blood. For experiments requiring a second fluorescent tag, Cy7-conjugated antibodies will be used which will be excited by a Ti:sapphire laser tuned to 745 nm. The stationary beam from Laser 2 will be focused by a cylindrical lens (L6) onto and optically opaque material located at the back focal plane of the objective lens L6. The optically opaque material will have a 0.3X3 mm slit aperture (S1) through which the stationary

beam will pass. After passage through S1, the stationary beam will be directed into the 30X, 0.9 NA water immersion objective lens. This lens will produce a 10X100 micron image of the slit S1 onto the blood vessel from which data will be acquired. The water immersion objective lens will orient the image of the slit such that the 100 micron dimension is perpendicular to the direction of blood flow (i.e. spans the diameter of the blood vessel), and the 10 micron is parallel to the direction of blood flow. (Note that it is being assumed here that the blood vessel from which data will be acquired has a diameter of 100 microns. If this is not the case then the larger dimension of slit S1 will be changed to 30D (where D = diameter of the blood vessel) so that the image produced by the water immersion lens spans the diameter of the blood vessel.) Consequently, no matter where the fluorescently tagged cells are located radially in the blood vessel, a burst of fluorescence photons will be generated when the fluorescently labeled cells flow by the location of data acquisition . (Note that having an image dimension parallel to the blood flow which has a magnitude comparable to the diameter of the cells will result in a high probability that only one fluorescently tagged cell will be in the probe volume at any one time.) The fluorescence will be collected by the water immersion objective lens and pass through an extraneous-light blocking slit aperture (S2), to a detector, where it will be recorded. If a second fluorescent label is being used then a laser-optics-detector setup similar to the Laser 2 system setup will be present. At times, video data of the fluorescently labeled cells will be obtained simultaneously with the fluorescent signal data. To accomplish this, Laser 1 and Laser 2 will both irradiate the specimen at the same time, and detector 1 will be interfaced with a video recorder (i.e. output of detector 1 will be fed into a video recorder). The pinhole P of the confocal microscope will be adjusted to have a confocal parameter of 100 microns so that only electromagnetic energy originating or backscattering from within the 100 micron diameter blood vessel reaches detector 1.

Probe laser power and tissue temperature The detection sensitivity of the above system will be a strong function of the power of the stationary probe beam. Obviously, the more powerful the probe beam, the more photons that will be present in the probe volume, the greater the detection sensitivity of the system. However, due to possible tissue damage resulting from elevated tissue temperature, there are limitations on the laser power of the stationary beam. Therefore, we propose to develop a new noninvasive optical temperature probe (i.e. technique)

that will interface with the in vivo flow cytometer and measure the temperature of the tissue in the probe volume. This will allow for the largest probe laser power that can safely be used during data acquisition, which may last for tens of minutes or even longer, depending on the scarcity of circulating tumor cells.

The method proposed here to determine soft tissue temperature is by analysis of the Raman scattering signal from the water vibrational band at 3400 cm^{-1}. This signal is well separated from other tissue Raman bands, and is roughly composed of two components centered near 3200 cm^{-1} and 3600 cm^{-1}. As the temperature of the soft tissue (and, hence, water) increases, the hydrogen bonding network becomes increasingly randomized, resulting in a Raman lineshape that changes with temperature (see Figure 2). Consequently, the ratio of the vibrational component at 3200 cm^{-1} to the vibrational component at 3600 cm^{-1} becomes a unique function of temperature, allowing determination of the soft tissue temperature. The Raman scattering signal will be generated by the probe beam itself (via inelastic scattering of the light by the water molecules in the soft tissue) and detected in the same slit geometry as the fluorescence burst, so that the temperature rise at the actual location where the probe beam intersects the tissue will be determined.

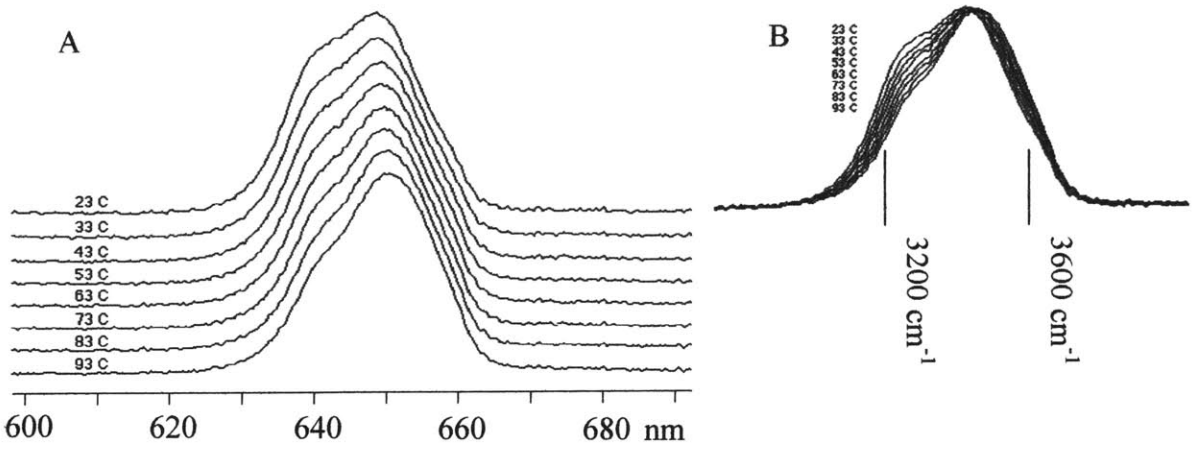

Fig. 2. Raman spectrum of the water O-H stretching band, excited with a frequency-doubled Nd:YAG laser at 532 nm. (A) Lineshape changes as the temperature increases from 23 C (top) to 93 C (bottom) in 10 degree increments. (B) Overlay plot of the same curves as in (A), showing that the low frequency (tetrahedral) component decreases while the high frequency (random network) component increases with increasing temperature.

The inelastically scattered light from the soft tissue can be imaged onto the entrance slit of a spectrometer, such as an Oriel Instaspec IV, and the signal converted to temperature. (Calibration of the device so that the spectrometer signal is converted to the correct soft tissue temperature can readily be performed prior to the experiment. In addition, performance of the probe will be checked against an approximate numerical model of the data acquisition situation by the FEM code ADINA-T to alert to any erroneous readings by the probe.)

Detection circuitry The circuitry required to interpret the fluorescent signals will need to be designed in the lab. This is due to the fact that the flow velocity and the radial location of the fluorescently tagged cells will be highly variable in the blood vessels. In addition, diffraction of light due to blood and soft tissue could noticeably affect the probe volume shape and, consequently, the intensity of the laser beam over the probe volume. Thus, the pulse widths, pulse shapes, and pulse peaks will be highly variable. Therefore, to design and build the required circuitry, we will first need to determine these parameters in vivo. To do this, the recorded fluorescent electrical signal data will be fed directly into the audio channel of a digital video camcorder. (This is possible because a digital video camcorder such as a Cannon ZR-10 has two built-in 16 bit digitizers (two audio channels) which each sample at 48 kHz.) Simultaneously, a recorded video signal obtained by the scanning confocal microscope will be fed into the video channel of the camcorder so that the fluorescent electrical signal data (being fed into the audio channel) are time correlated with it. The digitized electrical signal data from the fluorescent labels will be viewed while simultaneously watching the video of the flourescently labeled cells (recorded by the scanning confocal microscope). The pulse height and shape of the fluorescent electrical signal will be inspected for unique features that are present only when a fluorescent label is present. The circuitry will then be designed to respond only when a unique feature (such as a minimum pulse amplitude) occurs in the electrical signal from the fluorescent light recording channel. In addition, other unique features in the electrical signal data will try to be discovered to indicate other aspects of the flow profile of circulating tumor cells, such as radial location in the blood vessel and velocity of the tumor cells. The flow velocity information could perhaps be obtained via their pulse widths, with longer pulse widths indicating slower moving tumor cells. (Note that one could also obtain flow velocity data by using two probe volumes located a known distance apart and dividing by the time

between the fluorescent signal obtained from a circulating tumor cell moving through the blood stream through each probe volume. However, this would require a more elaborate experimental setup since two spatially separated probe volumes would be needed.)

The velocities obtained will be compared with the velocities for leukocytes published in the literature. It is expected (though not known for certain) that the in vivo measured velocities for circulating tumor cells will be similar to that of the leukocytes. The specific reason that this is expected is that the erythrocytes (which will travel with the leukocytes and the circulating tumor cells), being particles which deform as they travel through the blood stream, are subjected to a net force that causes the erythrocytes to migrate toward the center of the flow stream [Munn, Melder, and Jain, 1996]. Because the erythrocytes will probably greatly outnumber the circulating tumor cells, as they do the leukocytes, a radial force resulting from collision with the circulating tumor cells will push the tumor cells, as it does the leukocytes, toward the blood vessel wall [ibid]. Consequently, the circulating tumor cells and the leukocytes should have similar velocities. Of course, if other forces (that are not acting on the leukocytes) are acting on the circulating tumor cells, or the tumor cells are much larger in number, or they are are clumped together as they travel through the blood stream, then the tumor cells may not have the same kinematic behavior as the leukocytes. The detection circuitry will be designed and built to try to extract information from the flow to answer these questions and the other cancer metastasis questions posed earlier.

Characterization of system performance Once the system is constructed, system performance characterization concerning detecting particles and cells in vivo will be performed. For this aspect of the project, six performance limits of the device will be determined.

1) Establish the lowest number of fluorescent microspheres that can be detected in the circulation in a 30- minute period. Fluorescent microspheres 6 microns in diameter will be injected into an anesthetized mouse at concentrations ranging from 10^2 to 10^5 microspheres per 100 μl injected volume. The concentration $10^2/100$ μl corresponds (in a mouse) to approximately 1 microsphere per 10^4 circulating leukocytes, and the concentration $10^5/100$ μl corresponds to approximately 1 microsphere per 10 circulating leukocytes. The device will

count the number of microspheres it can detect in vivo over a 30-minute period, determining the lowest concentration at which microspheres can be reliably detected. (Fluorescent microspheres might be needed to help answer various circulatory questions in which one would not want to use fluorescently tagged tumor cells.)

2) Establish the lowest number of fluorescent molecules per microsphere that can be detected in vivo. For this test, fluorescent microspheres with a known density of fluorochromes per particle will be injected into an anesthetized mouse, and in vivo measurements taken to establish the minimum number of fluorochromes per microsphere that can be reliably detected. The density range to be (initially) investigated will be from 10^2 to 10^5 fluorochromes per particle. In addition, mixtures containing microspheres which have different number of fluorochromes attached to them will be injected to determine the device's ability to detect particles with low fluorochrome density in the presence of particles with high fluorochrome density.

3) Establish the smallest ratio of two labeled cell populations that the system can accurately discern. To determine this system parameter, mixtures with varying ratios of microspheres labeled with two different fluorophores will be injected into anesthetized mice (i.e. each mouse will be injected with one mixture that has a unique ratio of the two different fluorophores). The ratio obtained from in vivo measurements will be compared with the known injected ratio to determine the smallest ratio of two labeled cell populations that the instrument can accurately discern.

4) Establish the lowest number of cells that can be detected while in the circulatory system in a 30-minute period. For this test, PEQ226.5, an antibody specific to LNCaP cells, will be conjugated to Cy5.5 fluorescent probe. These immunofluorescent conjugates will be used to label LNCaP cells, which will then be injected into anesthetized mice at concentrations ranging from 10^2 to 10^5 cells per 100 μl. The number of detectable labeled cells in circulation over a 30-minute period will be determined. In addition, the length of time that the injected cells stay in circulation will be determined. In vivo measurements will be taken at 0, 6, 24, and 48 hours after injection.

5) Establish the lowest concentration of fluorescent antibodies needed for intravenous injection. This will be accomplished by injecting unlabeled LNCaP cells into an anesthetized mouse, followed by injection of Cy5.5-conjugated PEQ226.5 antibodies. The antibody mixtures injected will vary in antibody concentration from 0.1 to 1.0 mg/kg body weight. In a related experiment, the length of time that the injected antibodies stay in the circulatory system will be determined. For this experiment, Cy5.5-conjugated PEQ226.5 will be injected into the mouse, followed by injection of LNCaP cells, and in vivo measurements taken at 6, 24, and 48 hours later.

6) Verify that injected antibodiel specifically label LNCaP cells (and that there is not nonspecific uptake by circulating leukocytes). For this aspect of system performance characterization, double labeling and two-channel fluorescence detection will be required. Specifically, Cy5.5-conjugated PEQ226.5 antibodies and Cy7-labeled antibodies will be coinjected into an anesthetized mouse. If specific antibody labeling is successful, then the PEQ226.5 antibodies will selectively label the LNCaP cells, and the Cy7-labeled antibodies will only label the circulating leukocytes. Consequently, all Cy5.5 positive cells will be Cy7 negative, and vice versa. However, if the PEQ226.5 antibodies are nonspecifically taken up by some of the circulating leukocytes, then double positive cells will be detected in the two channels, resulting in the conclusion that PEQ226.5 antibodies do not specifically label LNCaP cells. If this proves to be the case, a new labeling antibody which specifically binds only to LNCaP cells will need to be found.

In vivo studies with a mouse tumor model Once this last system performance test is made to be successful, quantitation (i.e. data acquisition) of circulating tumor cells before and after cancer treatment (such as PDT) will begin. (This portion of the studies will be done in collaboration with Dr. Tayyaba Hasan's group at Wellman Labs, who have an established prostate cancer model) For data acquisition before cancer treatment, the background number of exfoliated LNCaP tumor cells in circulation will be determined. For this aspect of the project, approximately $2X10^6$ LNCaP cells (human prostate cancer cell line) in 100 μl media in 100 μl Mitrigel will be injected into the prostatic capsule of six to eight week old male

SCID mice following a transverse incision in the lower abdomen. In approximately six weeks, LN metastases will occur in approximately 20% to 40% of the mice, with tumor volumes of 30 to 50 mm^3 being reached. In vivo flow measurement over the six week period will be performed twice a week following implantation of the LNCaP cells using Cy5.5-labeled PEQ226.5 to detect the number of circulating LNCaP cells. After six weeks of tumor growth and data collection, cancer treatment (PDT) will be performed on the tumor-bearing mice. Specifically, each mouse will receive 0.25 mg/kg of liposomal BPD-MA intravenously via a tail vein injection. One hour later, a laparotomy will be performed and the tumors will be exposed to light which has a wavelength of 690 nm, which will activate the cell-killing photosensitizer absorbed by the tumor cells. The source of the light will be a diode laser. The diode laser will be coupled to a 1-mm diameter fiber optic cable, and the output laser light of the fiber will be imaged into a circle by a microscopic objective. This circle of light will cover the whole tumor and some of the adjacent normal tissue area. A small mechanical vibration will be applied to the fiber optic during treatment to average out speckles induced in the coherent radiation by the multimode fiber. The delivered fluence and fluence rate will be 50 J/cm^2 and 50 mW/cm^2, respectively. In vivo flow cytometry will be performed twice a week following photodynamic therapy (PDT) using Cy5.5-labeled PEQ226.5 to detect the number of circulating LNCaP cells. Three weeks after completion of the cancer treatment, when the treatment has had enough time to completely manifest its results, the mice will be sacrificed by carbon monoxide asphyxiation, at which point tumor volume will be measured and metastatic colonies in the lymph nodes will be quantified. All the data acquired will then be analyzed to answer questions concerning time course of tumor cell shedding, metastatic potential of the tumor, tumor burden, and degree of correlation of CTC count with animal responsivity to tumor therapy.

An in vivo flow cytometer for noninvasive detection and quantification of circulating cells

Novak J, Georgakoudi I, Wei X, Prossin A, and Lin CP

Wellman Laboratory of Photomedicine, Massachusetts General Hospital, Harvard Medical School, Boston MA, 02114

Abstract

An in vivo flow cytometer has been developed that enables the noninvasive detection and quantification of circulating fluorescently-labeled cells in live animals. Signal from a cell population of interest is recorded as the cells pass through a slit of light focused across an artery or vein of interest. Confocal detection of the excited fluorescence enables continuous monitoring of labeled cells in the upper layers of scattering tissue, such as the skin. The device has been used to characterize the in vivo kinetics of red and white blood cells circulating in the mouse ear vasculature. Potential applications in biology and medicine are discussed.

Present methods to detect and count various types of cells within the blood stream involve extraction of blood from the patient or animal followed by ex vivo labeling and detection. For example, standard flow cytometry involves taking blood samples, fluorescent labeling of specific cell populations, and passing these cells in a single file through a flow stream[1]. The cells are interrogated by a light source (usually a laser) to determine the types and number of cells based on their fluorescence and light scattering signals. Another contemporary method is usage of a hemocytometer in conjunction with a microscope. This method involves manual counting of cells against a grid while viewing with a microscope. Although both methods are successful techniques, they provide only a single time sample. Consequently, if the cell population of

interest varies unpredictably and/or rapidly with time, it will be difficult to obtain a valid temporal population profile, since it will be difficult to know when to sample. In addition, with both methods, blood must be withdrawn for each time point, and there is a significant time delay between blood withdrawal and analysis results. The development of confocal and two-photon imaging techniques has allowed detection of static and circulating fluorescently labeled cells in vivo[2]. However, extraction of quantitative information about the number and flow characteristics of a specific cell population can be extremely tedious. In addition, the high velocity of flowing cells, especially in the arterial circulation, makes it difficult and sometimes impossible to track the cells, even when images are captured at video rates. To remedy these problems, we have constructed a flow cytometer with the capability to detect and quantify the number and flow characteristics of fluorescently labeled cells in vivo and over a continuous time period.

The underlying principle of operation of the in vivo flow cytometer is confocal excitation and detection of fluorescently labeled cells in circulation. A schematic of the experimental set-up is shown in Figure 1. The animal to be studied is anesthetized and placed on top of the stage with its ear adhered onto a microscope slide using glycerine. An artery or vein of appropriate diameter is identified by transilluminating the ear vasculature with a 520 nm light emitting diode and imaging it using a 40X, 0.6 numerical aperture, infinity corrected objective and an achromat lens onto a CCD camera. The 520 nm provides good contrast for vasculature imaging due to the high hemoglobin absorption coefficient at this wavelength range. A dichroic beamsplitter (BS1) is used to reflect the transmitted light towards the CCD. The microscope objective and achromat lens together provide 6X magnification. The field of view of the transillumination system is 800 X 1000 μm with a lateral resolution of 1.47 μm per pixel. From the CCD image displayed on a computer screen, an appropriate vessel is selected that is large enough to allow detection of a

significant number of cells but small enough so that the excitation slit traverses its width completely. Precise determination of data acquisition location is needed for temporal studies, since the measurements have to be taken at the same location over time for valid comparison of the data.

Light from a He-Ne laser is then focused into a slit by a cylindrical lens and imaged across the selected blood vessel using an achromat lens and the same microscope objective as the one used for the transillumination imaging setup. Red and infrared light sources are ideal for this system, as they provide good penetration depth through tissues. In addition, there are a number of flurescent probes excited in this wavelength range that are used routinely for cell labeling. The size of the slit at the focal plane of the sample is approximately 5x72 □m. The axial resolution (i.e. the full width at half maximum of the light slit onto the sample in the axial direction) is approximately 50 μm, a value chosen to match the vessels of interest. The sample is positioned so that the long dimension of the slit traverses the width of the blood vessel; thus, fluorescence can be excited as the labeled circulating cells of interest pass through the slit. The emitted fluorescence is collected by the microscope objective, directed through the dichroic beamsplitter BS1, reflected by a mirror and dichroic beamsplitter BS2, and imaged onto a 200x3000 μm mechanical slit, which is confocal with the excitation slit. This confocal arrangement eliminates light from out of focus fluorescent and scattering sources. Fluorescence is detected using a photomultiplier tube placed directly behind the mechanical slit, sampled at a rate of 100 KHz using a data acquisition card and displayed and stored onto a computer. A 675 nm (+/-25 nm) bandpass filter placed in front of the confocal slit prevents most of the backscattered excitation light from entering the detector. The power of the He-Ne laser at the blood vessel is approximately 600 μW.

The digitized signal is analyzed off-line using software developed on the Matlab platform. The time sequence is filtered using a moving average window to remove high frequency noise. Control measurements performed at the data acquisition location before any fluorescent labels are introduced in the blood stream are used to determine the noise statistics, and only fluorescent peaks that exceed the noise level are counted. The number of fluorescent peaks, along with the height and full width at half maximum (FWHM) of each peak are determined using in-house developed algorithms.

Figure 2 depicts typical data traces acquired from the mouse ear vasculature. Peaks correspond to fluorescence from circulating human red blood cells, which were isolated, labeled ex vivo with 0.1 mM DiD (a lipophilic dye which attaches to cellular membranes), and injected into the mouse circulation via the tail vein. Data were acquired from an artery and the corresponding vein to assess the instrument's capabilities to detect differences in the flow characteristics. As illustrated in Figure 3, we find that the fluorescent peaks detected in the vein are approximately twice as wide as the ones detected in the artery. This suggests that venous flow is slower than arterial flow, since the cells take longer time to pass through the excitation slit. Such differences in flow velocity are consistent with previous studies[3].

Our capability to quantify the number of fluorescently labeled circulating cells in a reproducible manner is demonstrated in Figure 4a. Measurements were recorded over a three day period from the same artery of a mouse injected with DiD-labeled human red blood cells, as described above. The mean and standard deviation of the number of cells per minute passing through the selected artery on a given day was estimated from three traces, two minutes in duration each. The number of detected circulating red blood cells remains constant, as expected[4]. The kinetics of circulating ex vivo labeled red blood cells can be contrasted to the kinetics of

mouse white blood cells labeled in vivo with a fluorescently-tagged antibody. Specifically, 250 □L of rat antimouse CD45 monoclonal antibody labeled with cy-chrome was injected at a concentration of 1 mg/kg through the tail vein of a 6-8 week old Balb/c mouse, anesthetized with a mixture of ketamine and xylazine (7:1 ratio). In vivo flow cytometry measurements were performed at 0.4, 1.4, 4.3 8.3 and 25 hours following the injection. When the antibody was introduced in the vasculature, it labeled the circulating white blood cells, which expressed the CD45 antigen on their surface. The increase in the number of fluorescently labeled white blood cells detected within the first 1.4 hours represents the kinetics of antibody binding. In contrast to the fluorescently labeled red blood cells, the number of labeled white blood cells decreases rapidly by approximately 75% within the first 8 hours. This is expected, since the circulation time of some white blood cell populations, such as neutrophils, is on the order of hours[5]. In addition, white blood cells become eliminated either by lysis or phagocytosis as a result of antibody binding[6]. The latter measurements also demonstrate one of the key advantages of this technique; the capability to sample repeatedly the same animal over short and long time periods to acquire information about dynamic changes of interest.

In summary, we report on the development of a new technique, which combines the concepts of standard flow cytometry and confocal detection to allow acquisition of flow cytometric information in vivo, without the need to extract a blood sample. We demonstrate that using this technique, we can quantify the number of fluorescently-labeled circulating cells in a reproducible manner. In addition, we can characterize the flow characteristics of these circulating cells by determining the FWHM of the detected fluorescence peaks. The ability to monitor in a quantitative way circulating cells in vivo offers a number of advantages. For example, we can observe the cell population of interest in its native environment, free of artifacts potentially

introduced by cell isolation and processing procedures required to perform a conventional flow cytometry measurement. Furthermore, we can follow the same cell population continuously and over long periods of time to examine the dynamic changes in the circulation of different types of cells. We are currently using this technique to measure the circulation lifetime of different tumor cells and to study the relationship between metastatic potential and circulation time. Ultimately we want to investigate the use of circulating tumor cell count for monitoring anti-tumor treatment outcome. In addition, we are using the in vivo flow cytometer to monitor leukocyte populations in the peripheral circulation in response to therpeutic manipulation such as antibody therapy. Following the kinetics of white blood cell depletion is important in tissue and organ transplanation and autoimmune diseases such as rheumatoid arthritis and AIDS. Thus, this technique offers a relatively simple means of implementing in vivo measurements which could have a significant impact in both the basic science and clinical arenas.

References

1) Shapiro, H.M., *Practical Flow Cytometry Third Edition* (Wiley-Liss, New York, 1995).
2) *Handbook of Biological Confocal Microscopy Second Edition*, Edited by James P. Pawley (Plenum Press, New York, 1995).
3) Chen Z, Milner TE, Srinivas S, Wang X, Malekafzali A, Martin JC, Nelson JS, "Nonivasive imaging of in vivo blood flow velocity using optical Doppler tomography," Optics Letters, Vol. 22, No. 14 (1997)
4) Thibodeau GA, Patton KA, *Anatomy and Physiology Fourth Edition* (Mosby, St. Louis, 1999).
5) Sunanda Basu, George Hodgson, Melissa Katz, and Ashley R. Dunn, "Evaluation of role

of G-CSF in the production, survival, and release of neutrophils from bone marrow into circulation," Blood 100 (3) 854-861.

6) Wulf GG, Luo K-L, Goodell MA, Brenner MK. "Anti-CD45-mediated cytoreduction to facilitate allogeneic stem cell transplantation," Blood 101 (6): 2434-2439.

References (without titles)

7) Shapiro, H.M., (Wiley-Liss, New York, 1995).

8) Edited by James P. Pawley (Plenum Press, New York, 1995).

9) Chen Z, Milner TE, Srinivas S, Wang X, Malekafzali A, Martin JC, Nelson JS, Optics Letters, Vol. 22, No. 14 (1997)

10) Thibodeau GA, Patton KA, (Mosby, St. Louis, 1999).

11) Sunanda Basu, George Hodgson, Melissa Katz, and Ashley R. Dunn, Blood 100 (3) 854-861

12) Wulf GG, Luo K-L, Goodell MA, Brenner MK. Blood 101 (6): 2434-2439.

Figure Captions

Figure 1: Schematic of the in vivo flow cytometer experimental setup. L1: condenser lens; OL: microscope objective lens (40X, 0.6 NA, infinity corrected); BS1-2: dichroic beamsplitters; AL1-3: achromats; CL: cylindrical lens; M1-4: mirrors; NDF: neutral density filter; BPF: bandpass filter.

Figure 2: Representative traces of fluorescently labeled human red blood cells flowing through an artery (A) and a vein (B) of the mouse ear. Traces were acquired after implementing a 50-point moving window averaging of the original data.

Figure 3: Histograms representing the number of peaks with a specific FWHM representing circulating DiD-labeled red blood cells per minute in an artery (black) and a vein (gray) of a mouse ear. Note the shift to higher FWHM values for the vein data, representing slow blood flow characteristics.

Figure 4: (A) The number of human red blood cells, labeled ex vivo and injected in the mouse circulation through the tail vein, flowing through a mouse ear artery remains constant for a period of three days, as expected. (B) In contrast, the number of white blood cells labeled in vivo with a fluorescently-tagged antibody, vary in a dynamic and rapid way from the time of antibody injection.

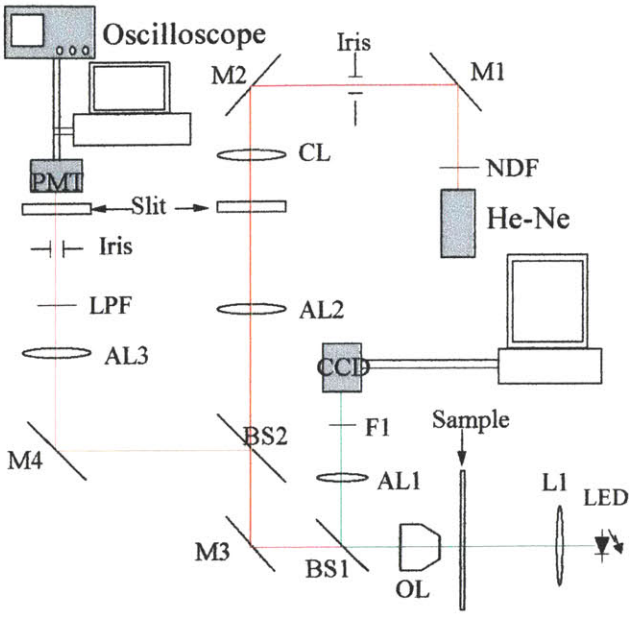

Figure 1.

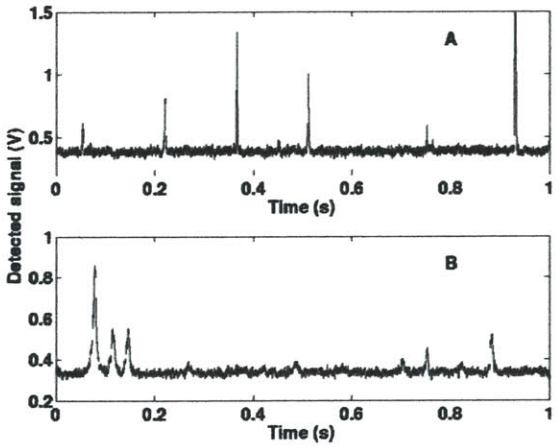

Figure 2.

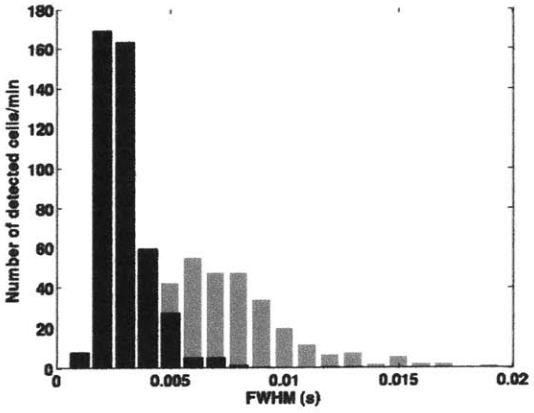

Figure 3.

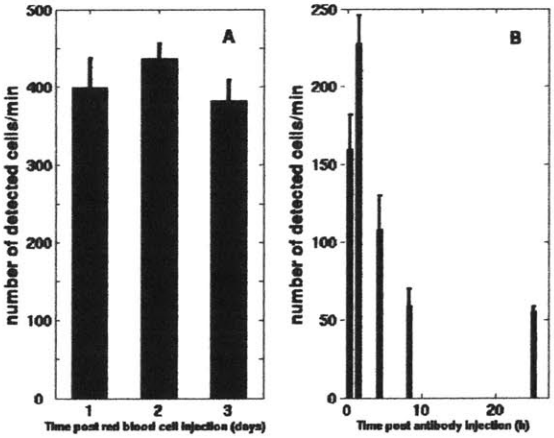

Figure 4.