FREQUENCY DEPENDENT INVESTIGATION OF THE VORTEX DYNAMICS IN HIGH TEMPERATURE SUPERCONDUCTORS

Thesis by

Daniel Seymour Reed

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1995

(Submitted January 18, 1995)

Acknowledgment

This thesis has been possible only with the help of numerous people with whom I have worked. I would first like to thank my thesis advisor, Prof. Nai-Chang Yeh, who has provided invaluable assistance and guidance to me in my experiments and preparation of this thesis and whose tireless devotion to our research has inspired our group to achieve all that we have done. I also could not have accomplished this work without the help and friendship of my fellow graduate students, Wen Jiang, Ushma Kriplani, and David Beam. I thank Nils Asplund who constructed the sample probes used in this work and whose technical expertise helped keep all of our vacuum systems and cryogenic systems functioning. I also thank all of the undergraduate students who provided assistance to our group, and in particular Amit Mehra and Jason Kumar. I would also like to thank all of the Caltech condensed matter physics group for making my time at Caltech the experience that it was.

I thank Dr. Fred Holtzberg who both supplied the $YBa_2Cu_3O_7$ crystals used in this work and hosted me for two visits to the IBM Thomas J. Watson Research Center. For hosting me for two months at Ecole Polytechnique in Palaiseau, France, and helping me learn about and perform the third harmonic susceptibility measurements presented in this thesis, I thank Dr. Marcin Konczykowski, Dr. Arkasha Samoilov, and Beat Schmidt. I also thank Dr. Mark Foote, Dr. Brian Hunt, Dr. Rick Vasquez and Dr. Jeff Barner at the Jet Propulsion Laboratory for teaching me about thin film growth and patterning as well as assisting us with sample preparation on numerous occasions.

I gratefully acknowledge a fellowship from the Schlumberger Foundation, and the financial support for this research from the David and Lucile Packard Foundation, the National Science foundation (grant #DMR-9401315), the Office of Naval Research (grant #N00014-91-J-1556), and NASA/OACT.

Above all others, I thank my parents for the support and encouragement they gave me during my stay at Caltech and helping me to accomplish all that I did which brought me here. Finally, I give my eternal gratitude to Pamela Speidel who drew many of the illustrations contained in this thesis, provided the moral support to help me through the writing of this work, and still agreed to become Pamela Reed. To my wife I dedicate this thesis.

Abstract

The high transition temperature and short coherence length of the high temperature superconducting cuprates lead to new vortex dynamics not found in conventional low temperature superconductors. The existence of a vortex-liquid state and a second-order phase transition between this vortex-liquid and the vortex-solid states has raised important questions about the vortex dynamics near this phase transition and about the interaction between the vortices and sample defects. In this thesis, new frequency dependent measurements systems are developed to measure the ac impedance and ac magnetic susceptibility of $YBa_2Cu_3O_7$ single crystals over a broad frequency range from 10^2Hz to 10^7Hz. In addition, a miniature Hall probe magnetometer is used to measure the third harmonic susceptibility of a $YBa_2Cu_3O_7$ single crystal. New critical scaling relations and analysis techniques are developed for ac magnetic susceptibility and the third harmonic transmissivity which for the first time enable critical scaling analysis to be applied to the measurements of these physical quantities. In twinned $YBa_2Cu_3O_7$ single crystals with only point defects, we provide the most conclusive evidence yet for the existence of a second-order vortex-solid to vortex-liquid phase transition by demonstrating consistent critical scaling for three different experimental techniques: dc voltage versus current, ac impedance versus frequency, and ac susceptibility versus frequency. We also use the new frequency-dependent techniques to investigate the effects of symmetry breaking in the vortex system by introducing columnar defects with heavy ion irradiation. In the case of parallel columnar defects, the symmetry is broken parallel and perpendicular to the columns, and the vortex behavior is consistent with a Bose-glass to vortex-liquid phase transition. If the

symmetry is broken further by introducing columnar defects at two different orientations, a new splayed-glass phase is found. The presence of canted columnar defects leads to stronger disorder and possible entanglement of the vortices resulting in slower critical dynamics near the vortex phase transition. The experimental techniques developed in this thesis provide valuable new tools for probing the vortex dynamics of these systems and future work on other vortex and spin systems.

Table of Contents

Chapter 1 : Introduction

Since the discovery of superconductivity in 1911, important technological accomplishments have been made in applications of superconducting materials. Superconducting magnets, SQUID detectors, and low noise microwave receivers are just a few of the applications that far outperform conventional technology.[1] However, the low temperatures required to operate these superconducting devices makes the cost of cooling the materials too high for most large scale applications. In 1986, the discovery of a new class of high temperature superconductors changed the prospect of large scale applications. A compound of Ba-La-Cu-O was found by Bednorz and Müller to be superconducting at a temperature of 30K.[2] Although 30K is only slightly higher than the former record of 23K for Nb_3Ge, the new discovery opened up superconductivity research into the new class of metal oxide ceramics. Soon after the initial discovery, materials with transition temperatures of 93K[3] and 110K[4] were found with the most recent record being 135K for $HgBa_2Ca_2Cu_3O_8$.[5]

The brittle nature and chemical reactivity of the high temperature superconducting materials have slowed the development of large scale applications of these materials, such as lossless transmission lines and superconducting magnets, but many small scale devices have been made from the high temperature superconductors. High temperature SQUIDs have been developed that operate at a temperature of 110K,[5] and superconducting microwave filters have been made to operate at 77K with properties superior to those of conventional copper filters at the same temperature.[6] Additionally, the large superconducting energy gap in the high temperature superconductors permits higher frequency applications for both analog signal processing and high speed electronics.[1]

In addition to important applications, the high transition temperatures of the new superconducting materials have raised many questions about the fundamental physics of

the high temperature superconductors. One issue that has received considerable attention is the very unusual "vortex dynamics" found in these materials. The high temperatures at which superconductivity exists gives rise to large thermal fluctuations and new vortex phases not seen in the low temperature superconductors. The nature of the new "vortex dynamics" found at high temperatures is investigated in this thesis, but before attempting to explain the meaning of "vortex dynamics," it is necessary to discuss some basic concepts of superconductivity.

1.1 The Superconducting State

The superconducting state is characterized by three features: zero electrical resistance, the ability to maintain a persistent electrical current in the absence of any elec-

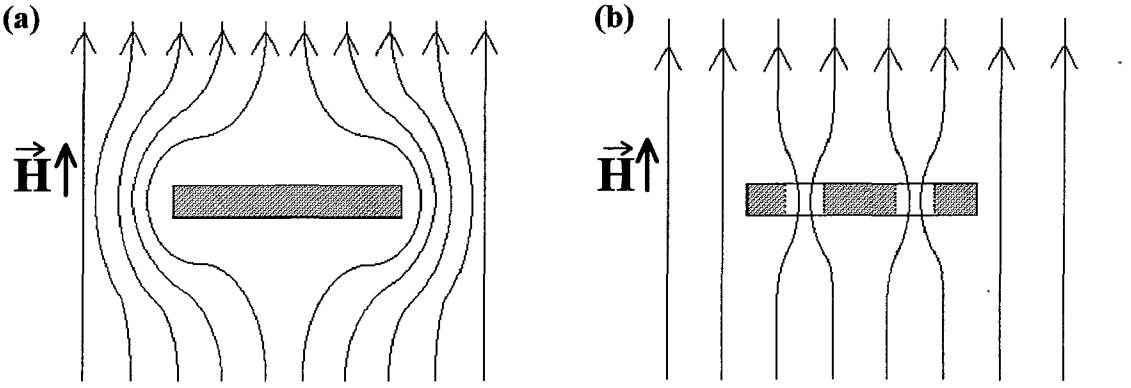
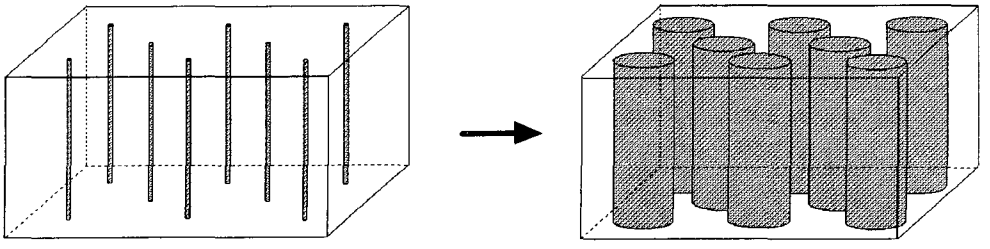


Figure 1.1. (a) Meissner state of a superconductor in which all magnetic flux is expelled. (b) Mixed state of a type-II superconductor in which magnetic flux penetrates only in quantized bundles.

tric field, and perfect diamagnetism. Zero electrical resistance just means that electric current can pass through the superconductor with no electric field generated along the sample and no electrical loss. With the absence of electrical resistance, the presence of persistent currents is understandable because there is no mechanism to reduce the current. If a current loop is created in a superconducting material, this current will continue to flow as long as the material remains superconducting. The third signature of superconductivity, perfect diamagnetism, is the expulsion of magnetic field from within the superconductor. This perfect diamagnetism is more than just a consequence of zero resistance and standard electromagnetic theory which asserts $\frac{\partial B}{\partial t} = -\nabla \times E$ where B is

the magnetic field and E is the electric field. A sample with zero resistance would always have E=0 so that $\frac{\partial B}{\partial t} = 0$ and the magnetic field would have to remain constant. The perfect diamagnetism found in superconductors leads to B=0 so that magnetic flux is actually expelled from the superconductor. This expulsion of magnetic field is illustrated in Figure 1.1(a) and is known as the Meissner state. For type-I superconductors, the Meissner state is the only state in which superconductivity can exist. If the applied magnetic field is increased beyond a critical field H_c, the superconductor immediately reverts to the normal (non-superconducting) state and the field penetrates the sample freely. However, for another class of superconductors (type-II) which includes the high temperature superconductors, for applied magnetic fields that exceed a lower critical field H_{c1}, the mixed state shown in Figure 1.1(b) is created. In this state, the magnetic field penetrates in discreet bundles made up of a single flux quantum, $\Phi_0 = \frac{hc}{2e} = 2.07 \times 10^{-7}$ gauss-cm^2. The region of penetration can be approximated by a core of normal material that is surrounded by a vortex of current that shields the rest of the superconductor from the presence of the local magnetic field. Because each vortex will contain exactly one flux quantum, the density of vortices must increase with increasing applied field until at the upper critical field (H_{c2}) the entire superconductor becomes normal.

1.2 The Vortex State



(a) Low Temperature **(b) Higher Temperature**

Figure 1.2. Vortex-solid in a low temperature superconductor. As the temperature increases to the superconducting transition T_{c2}, the vortex remains rigid but grows in diameter until the entire sample becomes normal at T_{c2}.

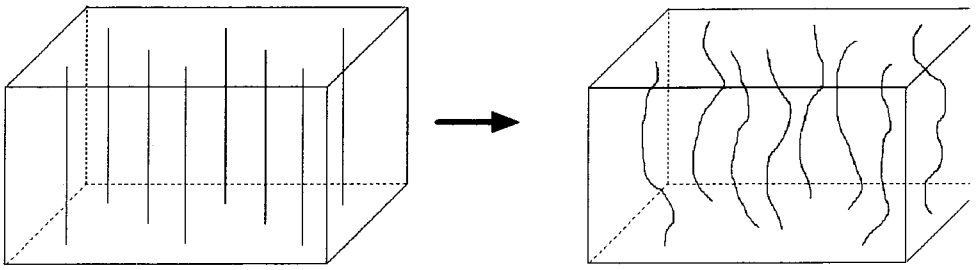


Figure 1.3. Vortex-solid melting in a high temperature superconductor. At low temperatures **(a)**, the vortices remain fixed in the vortex-solid, while at **(b)** higher temperatures, vortices are able to move.

In a fixed magnetic field H with $H_{c1} < H < H_{c2}$, the size of the vortex core is temperature dependent and grows with increasing temperature as indicated in Figure 1.2. For low temperature superconductors, the vortices remain essentially rigid lines that grow with increasing temperature until at the superconducting transition T_{c2} the entire sample becomes normal. However, in high temperature superconductors the situation is quite different as illustrated in Figure 1.3. For a temperature range below $T_{c2}(H)$, the thermal energy of the vortices becomes large enough that they are no longer rigid lines held in a "vortex-solid," but instead the "vortex-solid" melts into a "vortex-liquid" in which the vortices are able to move.

1.3 Measuring the Vortex Motion

In the vortex-solid state with no net vortex motion, the presence of a finite cur-

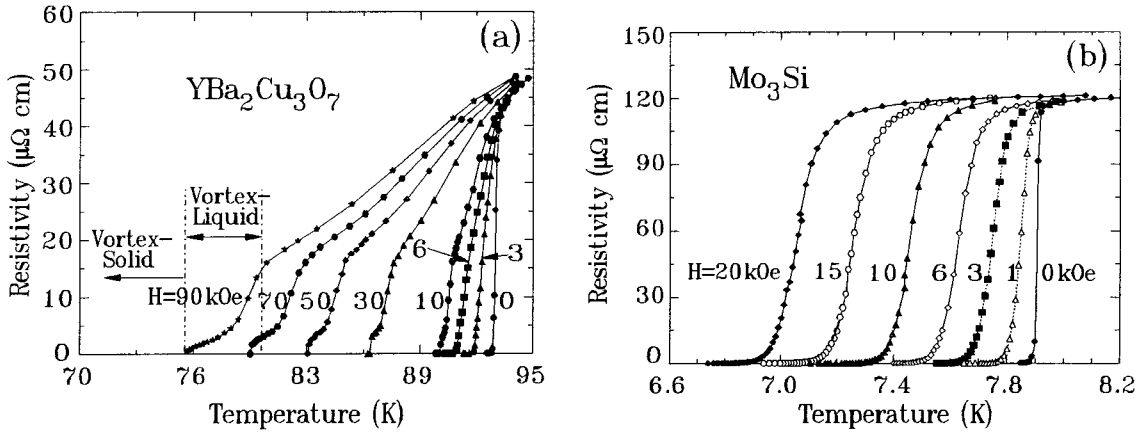


Figure 1.4. Plots of resistance versus temperature in different magnetic fields for **(a)** a high temperature superconductor and **(b)** a low temperature type-II superconductor. Note the broadening of the resistive transition in higher fields due to the presence of a vortex-liquid regime as indicated in **(a)** for H=90kOe.

rent density $\vec{j}$ below a certain critical current density J_c will not generate a detectable electric field $\vec{E}$ so the measured resistivity ρ will be effectively zero. The presence of the normal vortex cores has no effect on the measured resistance because the current follows the path of least resistance through the superconductor. However, the current applies a Lorentz force on the vortices given by $\vec{F} = \vec{j} \times \vec{B}$. If the vortices move at a velocity $\vec{v}$ in response to this force, the vortex motion creates an electric field given by $\vec{E} = \vec{B} \times \vec{v}$. For vortex motion $v > 0$, an electric field is generated parallel to the current and is measured as a finite resistivity. Thus, the resistance of the sample gives information about the motion of vortices. The plot of resistance versus temperature in Figure 1.4(b) clearly shows the enhanced resistance in the vortex-liquid regime in which the resistive transition is distinctly broadened in high magnetic fields. For comparison, Figure 1.4(a) shows a similar plot for a conventional low temperature type-II superconductor in which no vortex-liquid phase exists and the resistive transition remains sharp from the normal state to the vortex-solid state.

1.4 Thesis Overview

One key issue addressed in this thesis is whether there exists a true phase transition from the vortex-solid state to the vortex-liquid state as indicated in Figure 1.5, or just a gradual crossover. We investigate the dynamics of the vortex motion in the liquid state and near the transition. We also introduce controlled defects into the

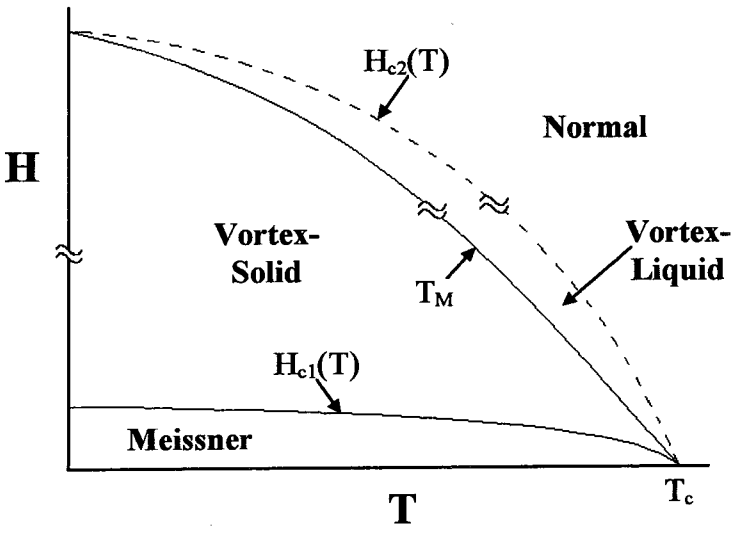


Figure 1.5. H vs. T phase diagram for high temperature superconductors showing the vortex-solid and vortex-liquid states separated by a phase boundary $T_M(H)$.

sample to see if the defects can "pin" the vortices and alter the dynamics. In order to investigate these issues, a variety of experimental systems are developed to probe the vortex dynamics. The measurement techniques presented here focus on studying the frequency dependence of the vortex response and include ac impedance measurements, ac magnetic susceptibility, and ac Hall probe magnetometry. In order to study the frequency dependence, it is necessary to perform the measurements over several orders of magnitude in frequency response, with 100Hz~10MHz chosen for ac impedance and susceptibility measurements. A large part of this thesis is dedicated to the techniques developed to overcome the experimental difficulties involved in measuring the low signal levels inherent in superconductors over this broad frequency range. Additional experiments have been performed at microwave frequencies[7] (10GHz to 18GHz) and found that at such high frequencies the vortex dynamics is dominated by the viscous motion of individual vortices.

This thesis begins by discussing some of the basic concepts and terminology of superconductivity as well as describing the conventional concepts of vortex dynamics such as flux creep and flux flow in Chapter 2. Chapter 3 introduces the concepts of a dislocation mediated melting and the critical phenomena which are needed to investigate the existence of a second-order vortex-solid to vortex-liquid phase transition and the nature of the vortex dynamics near this phase transition. After describing the theoretical background needed for this work, Chapter 4 describes some of the basic experimental techniques used for measuring the superconductors including sample preparation and general techniques used for cryogenic measurements. Chapters 5 through 8 are the highlight of this thesis which discuss the experimental techniques developed to measure the frequency dependence of the vortex response and the data obtained using each of the techniques. Chapter 5 describes the ac impedance measurements, Chapter 6 describes the ac magnetic susceptibility measurements, and Chapter 7 describes the ac Hall probe magnetometer. Chapter 8 describes measurements made on a unique sample that has

been irradiated to produce two sets of canted columnar defects. The measurements made on this sample emphasize the success of these measurement techniques and the importance of developing the frequency dependent measurements. Chapter 9 presents a summary of this work and discusses how these experimental techniques can be used together to thoroughly investigate the nature of the vortex-solid to vortex-liquid phase transition. Chapter 9 also summarizes the basic conclusions obtained from these experiments and how this work has led to a greater understanding of the vortex dynamics of high temperature superconductors.

A brief description and partial listing of the computer programs used to control the experimental systems are given in the appendices. Appendix A lists the primary routines used in performing the measurements described in this thesis. Appendix B lists the primary routines used in the microwave experiments discussed in reference 7.

References

[1] V. D. Hunt, *Superconductivity Sourcebook*, (John Wiley & Sons, Inc., New York 1989).

[2] J. G. Bednorz and K. A. Müller, Z. Phys. B **64**, 189 (1986).

[3] M. K. Wu, J. R. Ashburn, C. J. Torng, P. H. Hor, R. L. Meng, L. Gao, Z. H. Huang, Y. Q. Wang, and C. W. Chu, Phys. Rev. Lett. **58**, 908 (1987).

[4] M. A. Subramanian, J. C. Calabrese, C. C. Torardi, J. Gopalakrishnan, T. R. Askew, R. B. Flippen, K. J. Morrissey, U. Chowdhry, and A. W. Sleight, Nature **332**, 420 (1988).

[5] A. Gupta, J. Z. Sun, and C. C. Tsuei, Science **265**, 1075 (1994); C. C. Tsuei, A. Gupta, and G. Trafas, Science **263**, 1259 (1994).

[6] W. Chew, A. L. Riley, D. L. Rascoe, B. D Hunt, M. C. Foote, T. W. Cooley, and L. J. Bajuk, IEEE Transactions on Microwave Theory And Techniques **39**, 1455 (1991).

[7] N.-C. Yeh, U. Kriplani, W. Jiang, D. S. Reed, D. M. Strayer, J. B. Barner, B. D. Hunt, M. C. Foote, R. P. Vasquez, and A. Kussmaul, Phys. Rev. B **48**, 9861 (1993); U. Kriplani, N.-C. Yeh, W. Jiang, D. S. Reed, A. Gupta, and A. Kussmaul, *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p.795.

Chapter 2 : Superconductivity Background

Before any discussion of vortex dynamics can be pursued, it is necessary to understand some of the basic aspects of superconductivity and establish some of the terms used in the study of vortices and flux lines. This chapter is intended primarily to establish the terminology and basic concepts required for understanding the experimental work presented in subsequent chapters.[1]

2.1 What Is Superconductivity?

Superconductivity is more than just perfect conductivity with the resistance dropping to zero, but is a unique phase in the electronic structure that has one property of zero electrical resistance. A typical plot of resistivity (ρ) versus temperature (T) is shown in Figure 2.1 with the resistivity dropping to ρ=0 at the superconducting transition. Additionally, materials in the superconducting state are able to carry some finite current without any loss as long as the material remains in the superconducting state. Superconducting magnets take advantage of this persistent current by shorting the current leads with a superconducting shunt to hold a constant current through the magnet coil even after turning off the external current source. The other important feature of the superconducting state is the expulsion of magnetic fields from within the superconductor, or perfect diamagnetism.

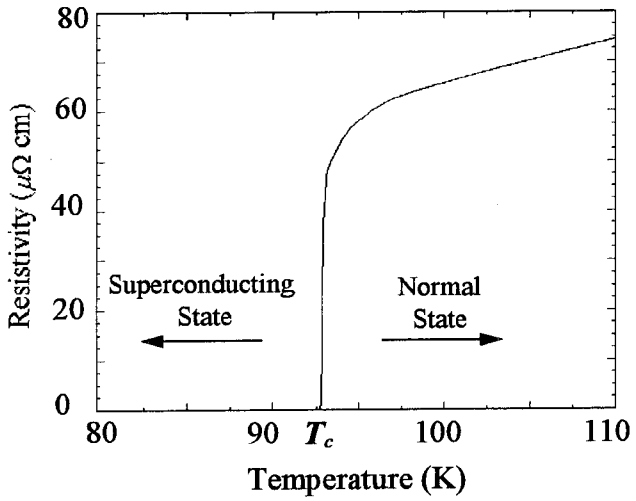


Figure 2.1. A plot of resistance versus temperature near the superconducting transition (T_c) for a $YBa_2Cu_3O_7$ single crystal.

2.2 Microscopic Properties

In order to understand the properties of a superconductor, it is helpful to consider the microscopic theory of superconductivity, BCS theory.[2] In the case of conventional low temperature superconductors, an interaction between the electrons and the phonons of the atomic lattice creates a bound pair of electrons called a Cooper pair. The electrons bound into the Cooper pairs are responsible for superconductivity. The binding energy of the Cooper pair creates a state in which each electron is at an energy Δ below the Fermi energy. This binding energy Δ is the superconducting gap energy, and as long as the kinetic energy of the paired electrons does not exceed the gap energy, the corresponding current will persist without dissipation. In order to understand the origin of the Meissner effect (perfect diamagnetism) in superconductors, it is useful to consider the canonical momentum of the electron in the presence of a magnetic field $\vec{p} = m\vec{v} + e\vec{A}/c$ where $\vec{A}$ is the vector potential of the magnetic field. In the absence of an electric field, the ground state should have zero net momentum $(\vec{p} = 0)$, so the average velocity is given by[3] $\langle \vec{v} \rangle = \frac{-e\vec{A}}{mc}$. In the case of normal metals, this argument does not apply because the electron velocity leads to an additional electric field, but in superconductors this electron velocity is responsible for the screening currents that expel the magnetic field. If the field gets too large and the kinetic energy of the Cooper pairs exceeds the gap energy, the pairs will break and the sample will become normal. For the case of type-I superconductors, this breakdown occurs at the critical field H_c and is equivalent to the field at which the diamagnetic energy exceeds the condensation energy. In another type of superconductors (type-II), the situation is slightly different, as will be described below, and the Cooper pairs are lost only when the applied field reaches the upper critical field H_{c2}.

A useful concept in superconductivity is the superconducting order parameter ψ. This order parameter is most rigorously defined through the Ginzburg-Landau equation for superconductivity,[1] but it is equivalent to the local density of superconducting electrons (Cooper pairs), $n_s(x) = |\psi(x)|^2$. As shown in Figure 2.2, neither the superconducting order parameter nor the magnetic field drops off immediately upon crossing a boundary from normal state material to superconducting material. Instead, the magnetic field drops off over a characteristic length scale λ and the superconducting order parameter drops off over a length of ξ_S. These two parameters, ξ_S and λ, are strongly temperature dependent and diverge to ∞ at the superconducting transition temperature T_c. However, they both diverge near T_c with a temperature dependence of $\xi_S(T) \sim \lambda(T) \sim |T - T_c|^{-1/2}$ so the ratio $\kappa \equiv \lambda/\xi_S$ is nearly constant, at least for $T \to T_c$. This κ is called the Ginzburg-Landau parameter and its value determines whether a superconductor will be type-I or type-II.

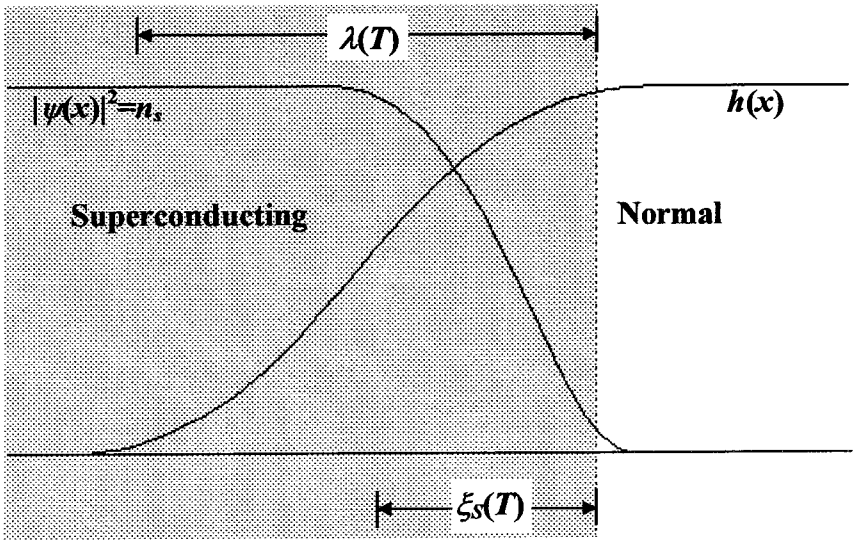


Figure 2.2. Interface between a superconducting domain and normal domain.

2.3 The Vortex State

For type-I superconductors with $\kappa < 1/\sqrt{2}$, at a fixed temperature $T<T_c$, as the applied magnetic field (H) is increased, the sample continues to expel the magnetic field until the diamagnetic energy exceeds the condensation energy and the sample returns to its

normal state. The transition from the superconducting state to normal state at $H_c(T)$ is shown in the phase diagram, Figure 2.3(a). This phase transition is generally first-order for bulk materials and becomes second-order in the thin film limit. However, for superconductors with $\kappa > 1/\sqrt{2}$ (type-II), another state exists.[4] For low magnetic fields, the superconductor remains in the Meissner state with all flux expelled, but for magnetic fields

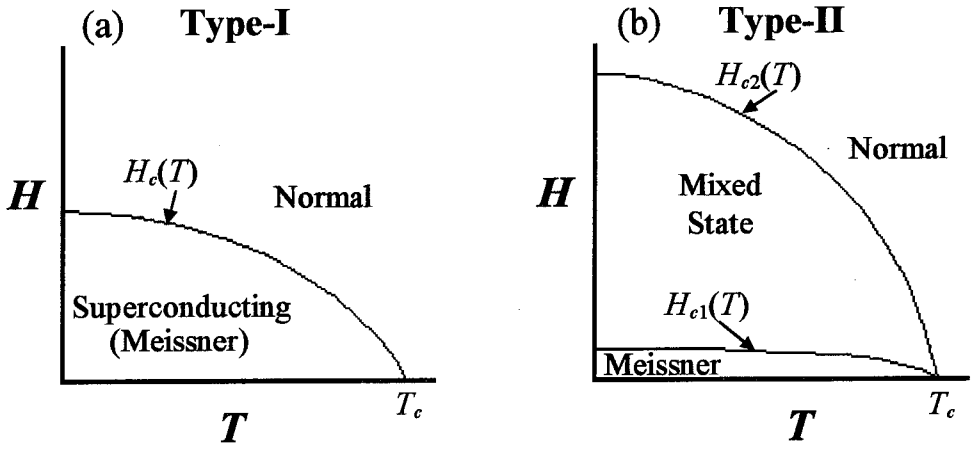


Figure 2.3. H vs. T phase diagrams for **(a)** a type-I superconductor and **(b)** a type-II superconductor.

greater than a lower critical field H_{c1}, a mixed state exists in which the magnetic field penetrates in small normal domains with most of the sample remaining superconducting. The mixed state was first proposed by Abrikosov[4] when he used the Ginzburg-Landau equations to show that a large κ leads to a negative surface energy between the normal and superconducting state. The negative surface energy leads to a subdivision of the normal and superconducting domains until each normal domain contains a single flux quantum $\Phi_0 = hc/(2e)$. As shown in Figure 2.4, the flux line penetrates the sample through a normal core of width ξ_S and is surrounded by a vortex of current that screens the magnetic field from the rest of the sample. Because the entire applied field H penetrates the sample in these quantized flux lines, the separation between vortex cores can be estimated as

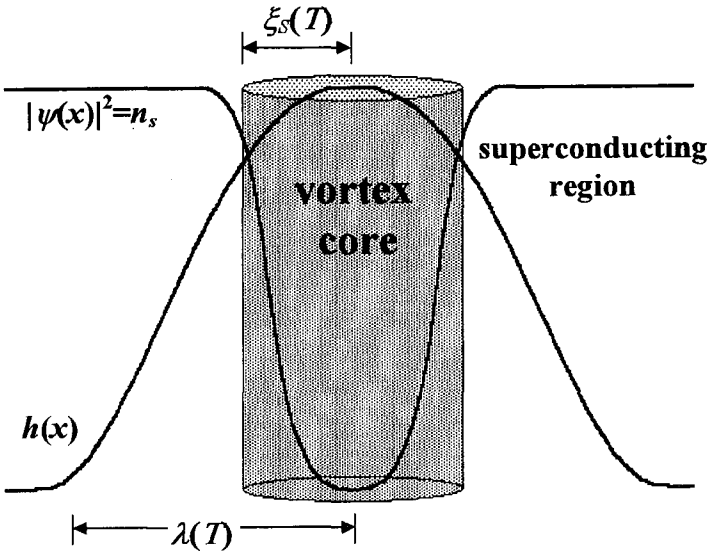


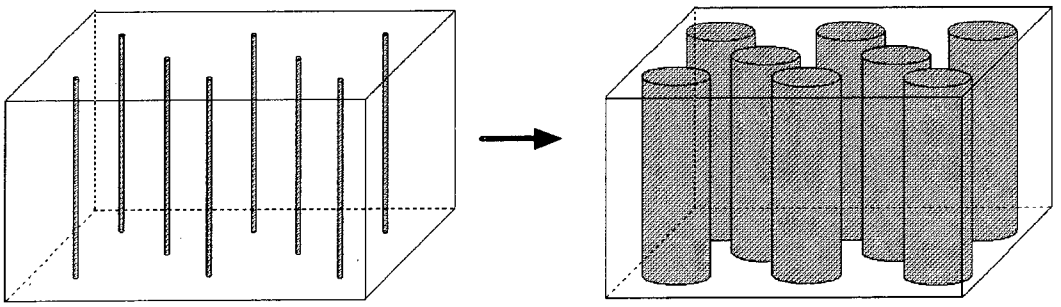
Figure 2.4. In the center of a vortex, the superconducting order parameter $\psi(x)=0$. The magnetic field $h(x)$ has a maximum at the center of the vortex and decays in the superconducting region with a characteristic length λ. The total flux through each vortex is a single flux quantum Φ_0.

$a = \sqrt{\Phi_0/B} \times 1.075$. The factor 1.075 is for a triangular lattice which is found to be the lowest energy configuration in a clean system.

The high temperature superconductors are type-II with a very large $\kappa \approx 200$ and small coherence length ξ_S. $YBa_2Cu_3O_7$ has a very anisotropic coherence length with $(\xi_S)_{ab} \approx 16$Å in the ab plane and $(\xi_S)_c \approx 2$Å along the c-axis,[5] but both values are far smaller than those found in conventional superconductors which for Nb is $\xi_S \approx 380$Å. The small coherence length makes the vortices in high temperature superconductors more sensitive to the effects of defects and sample disorder. Additionally, the very large κ leads to a small elastic energy between vortices.[6] In these large κ materials, the concept of discrete flux lines is misleading because the large magnetic penetration depth makes the magnetic field $h(x)$ nearly uniform over the sample. Even at fields as low as 1kOe, the change in field is <5% from the maximum at the vortex core to the minimum in the superconducting region. However, the position of the vortex core is clearly defined as the location at which $\psi \rightarrow 0$. The small coherence length, large κ, and high transition temperatures combine to give the high temperature superconductors unique features in the vortex dynamics.

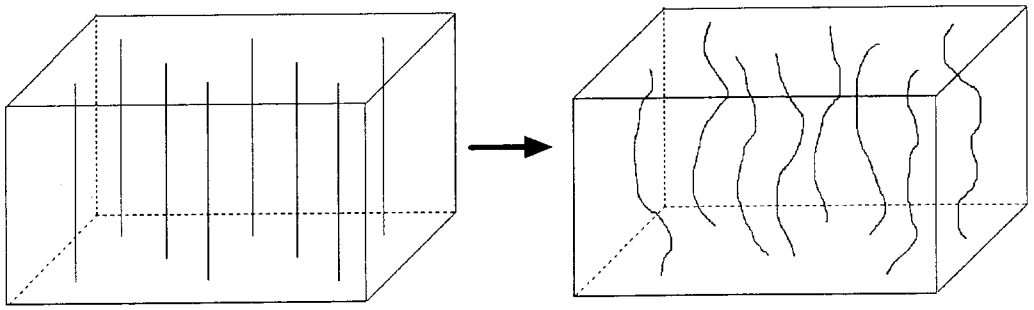
2.4 Vortex-Liquid

For conventional (low temperature) type-II superconductors, the large coherence length ξ_S makes fluctuation effects and sample disorder less important, and the low temperature provides little thermal activation to move the vortices. As illustrated in Figure 2.5, the effect of increasing the temperature is to increase the vortex diameter as $\xi_S(T)$ increases. The vortices remain essentially rigid lines up to the upper critical field $H_{c2}(T) = \Phi_0 / \left(2\pi\xi_S^2\right)$ at which point the entire sample becomes normal. However, in the case of high temperature superconductors, the vortices have a much smaller diameter due to the smaller ξ_S, and as the temperature increases, they can acquire sufficient thermal energy to overcome the elastic energy before reaching the upper critical field. The combination of the large thermal energy and short coherence length lead to a new vortex-liquid phase in which the vortex motion is similar to the motion of a viscous fluid. The existence of a vortex-liquid phase creates novel vortex dynamics in the high temperature superconductors.



(a) Low Temperature **(b) Higher Temperature**

Figure 2.5. Vortex-solid in a low temperature superconductor. As the temperature increases to the superconducting transition T_c, the vortex remains rigid but grows in diameter until the entire sample is normal at T_c.



(a) Low Temperature **(b) Higher Temperature**

Figure 2.6. Vortex-solid melting in a high temperature superconductor. At low temperatures **(a)**, the vortices remain fixed in the vortex-solid, while at **(b)** higher temperatures, vortices are able to move in the vortex-liquid state.

2.5 Vortex Motion

In its simplest form, the interaction between the flux lines and induced currents is quite easy to understand. By considering the Lorentz force $\vec{F} = \vec{j} \times \vec{\Phi}_0$, one easily sees that with a flux line in the $\hat{z}$ direction and current applied in the $\hat{y}$ direction, the Lorentz force will push the flux line in the $\hat{x}$ direction across the sample as shown in Figure 2.7. If the vortex is able to move, the Lorentz force gives rise to a vortex velocity $\vec{v} \| \vec{F}$, which in turn creates an electric field $\vec{E} \propto \vec{\Phi}_0 \times \vec{v}$ parallel to the applied current $\vec{j}$. The induced electric field is measured as a resistance in the sample and gives rise to energy dissipation.

In a perfectly clean system, any applied current should lead to vortex motion and cause finite resistance. However, any real sample has defects that act as pinning sites to hold the flux lines in place. The reason a sample defect acts to pin a flux

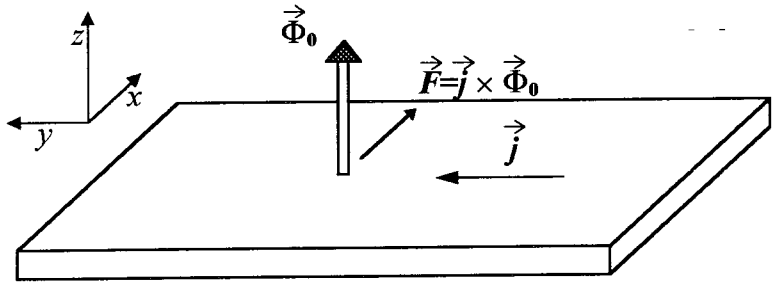


Figure 2.7. The Lorentz force $\vec{F} = \vec{j} \times \vec{\Phi}_0$ which acts to push the vortex across the sample under an applied current j.

line is because, by definition, a defect is a location in the sample at which the superconducting energy gap is smaller. The smaller gap at a pinning site implies a smaller condensation energy so that the flux line pays less penalty in energy by making the defect normal than by making a good part of the sample normal. The lower energy associated with keeping the vortex at the defect leads to a pinning potential $U(T)$ which decreases with increasing temperature because of the decreasing energy gap Δ, and vanishes at T_{c2}, where $T_{c2}(H)$ is defined as the temperature at which $H_{c2}(T)=H$.

2.5.1 Flux Flow

If vortex pinning is very weak or if sufficiently high current can be applied to the superconductor, the Lorentz force will be able to overcome the pinning energy and move the vortices. Treating the vortex motion in a purely phenomenological manner, the vortices encounter a viscous drag that can be described by a viscosity coefficient η. For the single flux quantum within the vortex, the Lorentz force per unit length is $j\Phi_0$, so for the vortex line moving at a velocity v_L, the equation of motion is simply $j\Phi_0 = \eta v_L$. Considering the electric field generated by vortex motion, a flux flow resistivity can be found to be $\rho_{ff} \equiv E/j = B\Phi_0 / \eta$ which is current independent. Using the Bardeen-Stephen model for the viscosity,[7] $\eta \approx \Phi_0 H_{c2} / \rho_n$, where ρ_n is the normal state resistance at the transition temperature, the flux flow resistivity is given by $\rho_{ff} \approx \rho_n B/H_{c2}$.

2.5.2 Flux Creep

At lower currents, a thermally activated flux creep process has been proposed by Anderson and Kim[8] which would also lead to a finite resistivity for temperatures $T>0$. The flux creep process takes the form of small bundles of vortex lines that occasionally have enough thermal energy to jump from one pinning site to another. The thermally induced jump rate is presumed to be of the form $R = \omega_0 \exp(-U / k_B T)$ where U is an activation

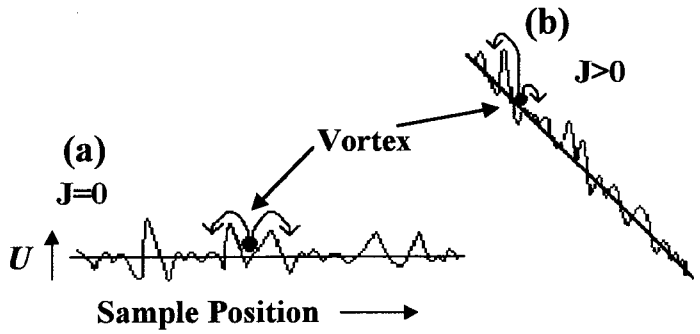


Figure 2.8. A representation of the pinning energy as a function of position across the sample and flux bundles jumping between adjacent sites. **(a)** Shows the motion with no driving force, and **(b)** shows the case of a driving current causing the flux bundles to favor "downhill" motion.

energy barrier that remains finite for any applied current, and ω_0 is a characteristic hopping attempt frequency. In the presence of an applied current, the energy barrier U is altered by the energy of the current as indicated in Figure 2.8. The change in energy is given by $\Delta U = jHV_cL$ where V_c is the volume of the flux bundle and L is the hopping distance. The effective energy barrier is U-ΔU in the forward direction and U+ΔU in the reverse direction which leads to a net creep velocity $v_L \propto \exp(-U/k_BT)\sinh(jHV_cL/k_BT)$. The creep velocity generates an electric field given by

$$E_{fc} = E_0 \exp\left[-U/k_BT\right]\sinh\left[jHV_cL/k_BT\right], \tag{2.1}$$

where E_0 is a constant coefficient. Equation (2.1) always assumes a finite pinning potential U and implies a finite resistivity at all temperature T>0 and current, which implies that no truly superconducting state exists.

2.5.3 Collective Vortex Behavior

The conventional Anderson-Kim flux creep model was at one time the only framework for understanding vortex motion and appeared to exclude the possibility of a rigid "vortex-solid" that would have no vortex motion. However, the conventional flux creep description assumes essentially non-interacting vortex bundles and neglects collective vortex correlations. Recent work that includes the collective effects of vortex interactions[9] suggests that the resistivity in Eq. (2.1) could take the form $E/j \sim \exp\left(-U/k_BT\right)$

with $U \sim \left(J_0/j\right)^{\mu}$ where J_0 is a characteristic current density. At lower temperatures, $\mu>0$ which leads to an **infinite** pinning potential at a finite temperature as $j \to 0$. The infinite potential creates a new vortex-solid phase in which no vortex motion occurs and can be described by a finite shear modulus C_{66}. The existence of a vortex-solid state has tremendous practical importance in providing a true zero resistance. The vortex-solid state also raises many questions about the fundamental physics and the nature of the vortex dynamics. One question of particular interest is whether there exists just a gradual crossover from the vortex-solid to the vortex-liquid, or if there exists a true phase transition. These issues will be addressed in the following chapters.

References

[1] For a more comprehensive description of superconductivity, see for example M. Tinkham, *Introduction to Superconductivity* (Robert E. Krieger Publishing Company, Inc., Malabar, FL, 1980).

[2] J. Bardeen, L. N. Cooper, and J. R. Schriever, Phys. Rev. **108**, 1175 (1957). For another description of BCS theory, see also Ref. 1.

[3] F. London, *Superfluids*, Vol. 1 (Wiley, New York, 1950).

[4] A. A. Abrikosov, Zh. Eksperim. i Teor. Fiz. **32**, 1442 (1957) [Soviet Phys.–JETP, **5**, 1174 (1957)].

[5] B. Oh, K. Char, A. D. Kent, M. Naito, M. R. Beasley, T. H. Geballe, R. H. Hammond, and A. Kapitulnik, Phys. Rev. B **37**, 7861 (1988).

[6] E. H. Brandt, Phys. Rev. Lett **63**, 1106 (1989).

[7] J. Bardeen and M. J. Stephen, Phys. Rev. **140**, A1197 (1965).

[8] P. W. Anderson, Phys. Rev. Lett. **9**, 302 (1962); P. W. Anderson and Y. B. Kim, Rev. Mod. Phys. **36**, 39 (1964).

[9] M. V. Feigelman, V. B. Geshkenbein, A. I. Larkin, and V. M. Vinokur, Phys. Rev. Lett. **63,** 2303 (1989).

Chapter 3 : Second-Order Vortex Phase Transition and Critical Phenomena

The existence of a vortex-solid state at low temperatures with a non-zero shear modulus ($C_{66}\neq 0$) is important in establishing a truly superconducting state with zero resistivity. However, a more interesting physics issue is what happens as the temperature is increased and the shear modulus becomes zero in the vortex-liquid state. Instead of a gradual crossover, there could exist a true phase transition separating the vortex-solid from the vortex-liquid as shown in Figure 3.1. Some experimental signatures of a second-order vortex-solid to vortex-liquid transition have been discussed theoretically in papers by Fisher, Fisher and Huse,[1] by A. T. Dorsey,[2] as well as by Nelson and Vinokur.[3] This chapter discusses how the critical scaling hypothesis is used by us[4-14] to investigate the vortex dynamics near the melting transition and the experimental signatures associated with a second-order vortex-solid to vortex-liquid phase transition.

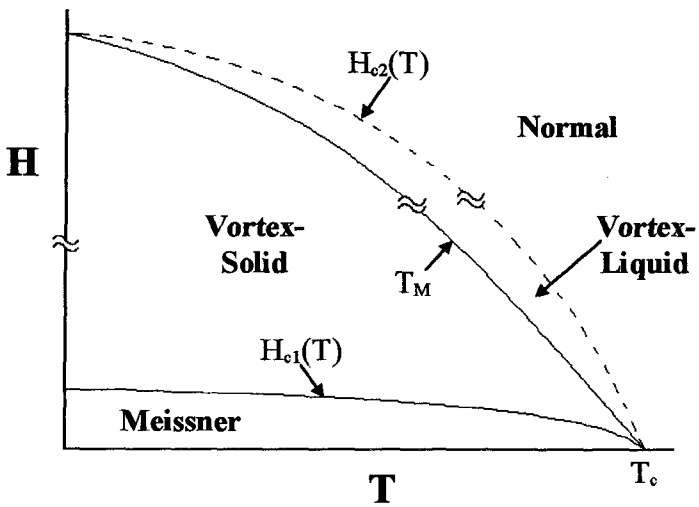


Figure 3.1. H vs. T phase diagram for high temperature superconductors showing the vortex-solid and vortex-liquid states separated by a phase boundary $T_M(H)$.

3.1 Correlation Length and Critical Exponents

The theory of critical phenomena is a general theoretical framework that describes the behavior of a system near a second-order phase transition. One of the central features

of critical phenomena[15] is the existence of a diverging correlation length $\xi_V = \xi_0|1-T/T_M|^{-\nu}$ for temperatures near the transition temperature T_M, where $\nu>0$ is the static exponent. The vortex correlation length ξ_V should not be confused with the superconducting coherence length ξ_S which is related to the superconducting order parameter ψ and diverges only at the superconducting transition temperature T_c as $\xi_S \sim |1-T/T_c|^{-1/2}$.

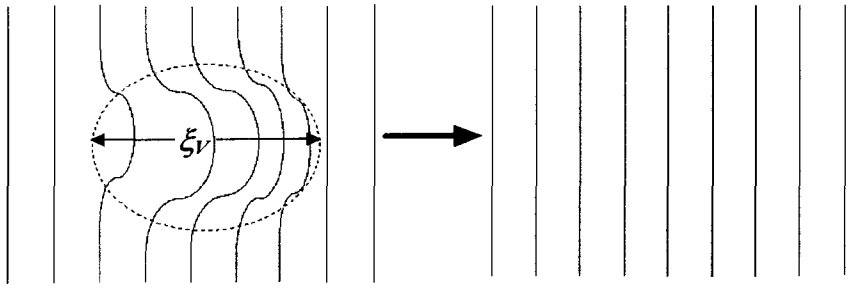


Figure 3.2. (a) Illustration of the vortex dislocation loop which after a time τ returns (b) to the ground state.

In the vortex-solid phase ($T<T_M$), ξ_V is the size of dislocation loops that form as shown in Figure 3.2(a), and in the vortex-liquid phase ($T>T_M$), ξ_V is the length over which vortices are spatially correlated. We note that the dislocation loop is a three-dimensional object because the vortex motion is correlated in the direction out of the page as well as the two directions indicated in Figure 3.2(a). Although there are many length scales involved in determining the vortex dynamics, most length scales vary slowly near T_M, with length scales such as ξ_S and λ diverging only at T_c. Only ξ_V diverges near T_M, so for temperatures within a narrow critical regime about T_M, this diverging correlation length will be the most important length scale and will govern the properties of the vortex system.

The theory of critical phenomena further asserts[15] that in the limit as the temperature T approaches the transition temperature T_M, relevant physical quantities may scale with the correlation length. Thus, the characteristic relaxation time $\tau(T\rightarrow T_M)$ is given by $\tau \sim \xi^z$ where z is the dynamic exponent. The relaxation time τ is the time required for

fluctuating vortex dislocations to return to the unperturbed state, as indicated in Figure 3.2. The divergence of ξ_V implies that τ will also diverge at T_M, making τ the most important time scale for the vortex dynamics. The physical meaning of the relaxation time τ may be related to the thermal relaxation rate of the elastic response[8, 16]

$$\tau^{-1} \sim \frac{C_{66} k_{\perp}^2}{\eta} \tag{3.1}$$

where η is the viscosity, Φ_0 the flux quantum, C_{66} the shear modulus of the vortex-solid, and $k_{\perp}$ the transverse wave vector of the vortex-solid. Although Eq. (3.1) is not a rigorous description of the time scale τ, it shows the consistency of the diverging time scale ($\tau \rightarrow \infty$) with the vanishing shear modulus, $C_{66} \rightarrow 0$.

3.2 Scaling Relations

3.2.1 AC Impedance

In order to investigate the critical phenomena near a second-order vortex phase transition, it is convenient to find scaling relations to describe how the measured quantities behave near T_M. We may first define a characteristic frequency as $f_T = \tau^{-1} \sim \xi_V^{-z}$. Next we consider a characteristic current density J_T. The scaling current J_T is defined by the expression:[1, 17]

$$J_T \Phi_0 \xi_V^{d-1} = k_B T \tag{3.2}$$

where d=3 is the dimensionality of the phase transition. Equation (3.2) can be interpreted as comparing the work done on a dislocation loop by the Lorentz force and the thermal energy, so we find a characteristic current density $J_T = (k_B T/\Phi_0)\xi_V^{-2}$. As indicated in Figure 3.3, J_T separates the region in which thermal effects dominate from the region in which current effects dominate. Next we consider the sample resistivity $\rho(T,f,J)$ as a

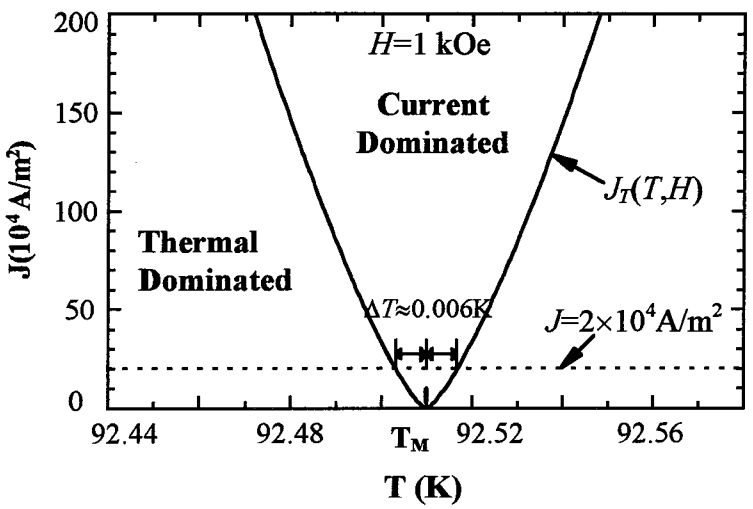


Figure 3.3. The current density (J) vs. temperature (T) diagram for H=1kOe. The solid line shows the characteristic current density J_T which separates the current dominated regime from the thermal dominated regime. The dashed line indicates the current density used for the ac impedance measurements in Chapter 5.

function of temperature (T), frequency (f), and applied current density (J). The resistivity is related to the superfluid density ρ_S by $\rho^{-1} \sim i\rho_S/f$. This ρ_S is not the usual superfluid associated with the density of superconducting carriers n_s, but it is a generalized superfluid density more accurately compared to the stiffness of the vortex-solid or the shear modulus. In the critical regime, the superfluid density scales as[18] $\rho_S \sim \xi_V^{2-d}$, where d is the dimensionality of the phase transition. Note that for d=3, $\rho_S \to 0$ at T_M, in contrast to $n_s \to 0$ at T_{c2}. Combining this expression with the frequency scaling $f \sim \xi_V^{-z}$, gives the scaling for the resistivity $\rho \sim \xi_V^{d-2-z}$. The scaling relation can be written more generally as

$$\rho(\delta; f, J) \sim \xi_V^{\,d-2-z}\,\tilde{\rho}_{\pm}\left(\delta\xi_V^{1/\nu}; f\xi_V^{z}; \frac{J}{J_T}\right), \qquad \delta \equiv \left|1-(T/T_M)\right|, \tag{3.3}$$

where $\tilde{\rho}_{\pm}$ is the universal function for temperatures $T>T_M$ (+) and $T<T_M$ (-). When considering a complex impedance $\rho = |\rho| e^{i\Phi}$, Eq. (3.3) can be written as $\rho \sim \xi_V^{d-2-z}\tilde{\rho}_{\pm} e^{i\tilde{\Phi}_{\pm}}$. With the applied current in the low current limit ($J << J_T$), Eq. (3.3) can be written more simply in terms of the scaled frequency $\tilde{f}$ by

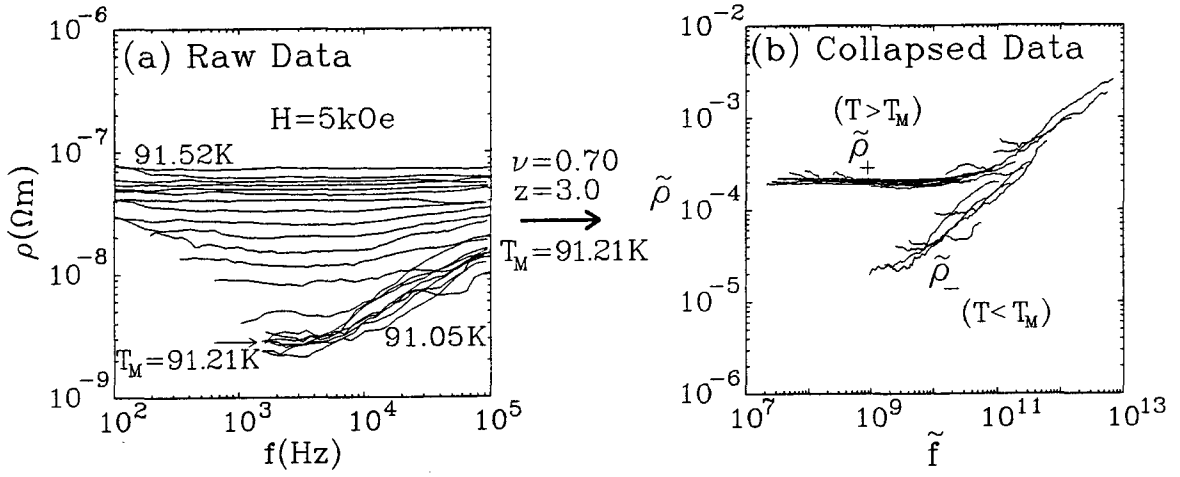


Figure 3.4. Impedance amplitude (ρ) versus frequency (f) isotherms which are "collapsed" into the universal function $\tilde{\rho}_{\pm}$ using Eq. (3.4) and ν=0.7, z=3.0, and T_M=91.21K.

$$\begin{aligned} \tilde{\rho}_{\pm}(\tilde{f}) &\equiv |\rho|\left|1-(\mathrm{T}/\mathrm{T_M})\right|^{-\nu(2-d+z)} \\ \tilde{\Phi}_{\pm}(\tilde{f}) &\equiv \Phi \\ \tilde{f} &\equiv f\left|1-(\mathrm{T}/\mathrm{T_M})\right|^{-\nu z}, \end{aligned} \tag{3.4}$$

where $\tilde{\rho}_{\pm}$ and $\tilde{\Phi}_{\pm}$ are the universal functions for $T<T_M$ (-) and $T>T_M$ (+). If the above scaling results are correct, then we can find values of ν, z, and T_M, which "collapse" the ρ vs. f into the pair of universal functions $\tilde{\rho}_{\pm}(\tilde{f})$ as shown in Figure 3.4. In the limit of $T\to T_M$, $\rho(f)$ will follow the same power law as $\tilde{\rho}(\tilde{f})$, so that $\rho(f, T\to T_M)\propto f^x$ where $x=1-(d-2)/z$. By inserting the phase and amplitude of the resistivity into the Kramers-Kronig relation,[2]

$$\Phi=-\mathcal{P}\int_{-\infty}^{\infty}\frac{df'}{\pi}\frac{\ln|\rho(f')|}{f'-f} \quad\to\quad \Phi(T_M)=\frac{\pi}{2}\left[1-\frac{d-2}{z}\right] \tag{3.5}$$

where $\mathcal{P}$ denotes the principal value of the integration. Equation (3.5) implies that for $T\to T_M$, the phase becomes frequency independent and the value of $\Phi(T_M)$ provides an additional way to verify the value of the dynamic exponent z.

3.2.2 AC Magnetic Susceptibility

The first derivation of the scaling relation for magnetic susceptibility was derived incorrectly beginning with the susceptibility in terms of the free energy density[1] f_S, $\chi \equiv \frac{\partial^2 f_S}{\partial H^2}$. Using $f_S \sim \xi_V^{-d}$ and $H \sim \xi_V^{-2}$, the susceptibility becomes $\chi \sim \xi$ which implies a susceptibility diverging at the transition temperature T_M, a physically unreasonable prediction! Although the expression for susceptibility derived in reference 1 may be valid for systems such as spin systems, in the case of a superconductor, the free energy density of the vortices does not scale trivially with the volume density ξ_V^{-d}, and it is necessary to consider the origin of the susceptibility signal. The susceptibility signal comes from induced screening currents in the sample which penetrate a depth of λ, the dynamic penetration depth. In the linear response limit,[6] $|\chi| \sim (a/\lambda)$ where a is a characteristic sample dimension and λ follows the scaling relation[1, 18] $\lambda \sim \rho_S^{-1/2} \sim \xi_V^{(d-2)/2}$. Combining the expressions for λ and χ and using d=3, we obtain $\chi \sim \xi_V^{-1/2}$ which gives $\chi(T \to T_M) \to 0$ in agreement with experimental observations[6] (see Chapter 6). Defining the phase of the susceptibility by $\chi = |\chi| e^{i(\pi - \phi_\chi)}$, the scaling relations obtained for the susceptibility amplitude and phase are given by:[6]

$$\begin{aligned} \tilde{\chi}_{\pm}(\tilde{f}) &\equiv |\chi| \left|1 - (T/T_M)\right|^{-\nu/2} \\ \tilde{\phi}_{\chi\pm}(\tilde{f}) &\equiv \phi_\chi \\ \tilde{f} &\equiv f \left|1 - (T/T_M)\right|^{-\nu z} \end{aligned} \tag{3.6}$$

with $\tilde{\chi}_{\pm}$ and $\tilde{\phi}_{\chi\pm}$ being the universal scaling functions for the amplitude and phase, and the "±" defined for $T > T_M$ (+) and $T < T_M$ (-). Using the Kramers-Kronig relation, we obtain an expression similar to Eq. (3.5) for the phase of the susceptibility,[2, 8]

$$\pi - \phi_\chi(T \to T_M) = \pi - \frac{\pi}{2}\left(\frac{1}{2z}\right), \tag{3.7}$$

which is a frequency independent phase at the transition T_M.

3.3 Symmetry Breaking

The previous scaling relations have all been derived based on the assumption of an "isotropic system" in the sense that the correlation lengths in all directions have the same temperature dependence near the second-order phase transition. In other words, if we define the vortex correlation lengths in all three directions to be $\xi_i = \xi_{i0}\left|1-\left(T/T_M\right)\right|^{-\nu_i}$ where i=x, y, z, then ν_x=ν_y=ν_z. Now we consider the consequences of breaking the isotropic symmetry to see how the scaling relations are modified.

Considerable attention has been given to superconducting crystals that have been irradiated with high energy ions to produce long columns of amorphous material through the sample.[3, 12, 19-22] As illustrated in Figure 3.5, the columns create long pinning centers which can interact with a vortex along its entire length. Nelson and Vinokur[3] have shown that the interaction between flux lines and columnar defects can be mapped to the quantum mechanics of two-dimensional Bosons on a disordered substrate with the direction of columns being an imaginary time axis. Although the scaling relations for the "Bose-glass" to vortex-liquid phase transition were analyzed in the context of the two-dimensional Boson system, we consider a simpler approach of extending the scaling relations found above to an anisotropic system with $\nu_x \neq \nu_y \neq \nu_z$. We first consider a gen-

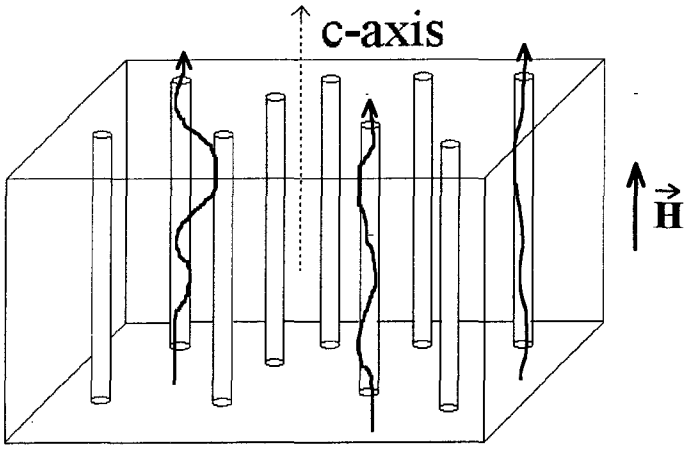


Figure 3.5. Illustration of how the flux lines interact with columnar defects in a superconductor.

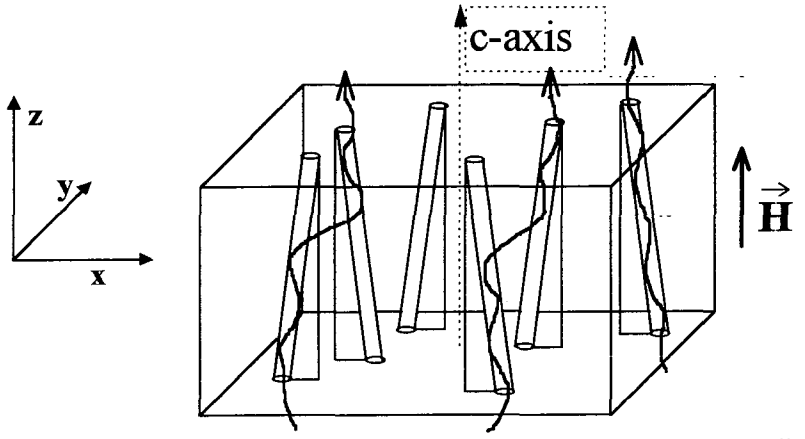


Figure 3.6. Configuration used to analyze the fully anisotropic scaling relations with canted columnar defects. Note that all columns are in the xz plane so the symmetry between the x and y directions is broken..

eralized system such as the one shown in Figure 3.6 in which the symmetry is broken in all three directions which includes the Bose-glass system as a special case with $\nu_x = \nu_y$. If, as indicated in Figure 3.6, the current is applied with $\vec{J} \| \hat{y}$, we expect the only relevant relaxation to occur in the x direction, parallel to the Lorentz force so the relaxation rate is given by $f_T \sim \xi_x^{-z'}$. Similarly, the elastic energy due to current induced distortion is $J\Phi_0\xi_x\xi_z$, so in analogy to Eq. (3.2), the characteristic scaling current becomes

$$J_T = \frac{k_B T}{\Phi_0 \xi_x \xi_z} \sim \frac{k_B T}{\Phi_0} \delta^{(\nu_x + \nu_z)}, \qquad \delta \equiv \left| 1 - T/T_G \right| \tag{3.8}$$

where T_G is the phase transition temperature. In the case of $\nu_x = \nu_y$, $T_G(=T_{BG})$ is the Bose-glass to vortex-liquid transition temperature, and for $\nu_x \neq \nu_y$, $T_G(=T_{SG})$ is the splayed-glass to vortex-liquid transition temperature. In order to find the scaling for the resistivity, we consider the superfluid density scaling $\rho_S \sim \xi_V^{2-d}$. Extrapolating this expression to the anisotropic system, we expect $\rho_S \sim \frac{\xi_x \xi_y}{\xi_x \xi_y \xi_z} \sim \xi_z^{-1}$ and find the resistivity scaling relation to be:

$$\rho^{-1} \sim i \frac{\rho_S}{f} \sim \frac{\xi_z^{-1}}{\xi_x^{-z'}} \sim \delta^{\nu_z - \nu_x z'}. \tag{3.9}$$

Rewriting Eqs. (3.8)-(3.9), we find the following scaling relations for the ac resistivity:

$$\rho(T,f,J\to 0)=\delta^{(\nu_x z'-\nu_z)}\tilde{\rho}_{\pm}\left(\tilde{f}\right),\qquad \tilde{f}\equiv f\delta^{-\nu_x z'} \tag{3.10}$$

where $\tilde{\rho}_+$ and $\tilde{\rho}_-$ denote the scaling functions for $T>T_G$ and $T<T_G$, respectively. The electric field (E) versus current density (J) isotherms in the dc limit ($f\to 0$) follow the scaling expression:

$$E(T,f\to 0,J)=\delta^{(\nu_x z'-\nu_z)}\tilde{E}_{\pm}\left(\tilde{J}\right),\qquad \tilde{J}\equiv\left(J/T\right)\delta^{-(\nu_x+\nu_z)},\qquad \tilde{E}_{\pm}=J\tilde{\rho}_{\pm} \tag{3.11}$$

We note that for current applied with $\vec{J}\|\hat{x}$, we get the same results except $\nu_x\to\nu_y$.

In the case of parallel columns discussed by Nelson and Vinokur,[3] $\hat{z}$ is defined to be the axis parallel to the columnar defects and the $\hat{x}$ and $\hat{y}$ directions are symmetric. The exponents can be redefined as $\nu_z\equiv\nu_\parallel$, $\nu_x=\nu_y\equiv\nu_\perp$, and the anisotropy ratio is defined by $\zeta\equiv\nu_\parallel/\nu_\perp$. Inserting these redefined exponents into Eqs. (3.10) and (3.11), we obtain the same results as those obtained for the Bose-glass transition:[3, 12]

$$\rho(T,f,J\to 0)=\delta^{\nu_\perp(z'-\zeta)}\tilde{\rho}_{\pm}\left(\tilde{f}\right),\qquad \tilde{f}\equiv f\delta^{-\nu_\perp z'},\qquad \delta\equiv\left|1-T/T_{BG}\right| \tag{3.12}$$

where T_{BG} is the Bose-glass to vortex-liquid transition temperature. Similarly, the electric field versus current density isotherms in the dc limit ($f\to 0$) follow the scaling expression:

$$E(T,f\to 0,J)=\delta^{\nu_\perp(z'-\zeta)}\tilde{E}_{\pm}\left(\tilde{J}\right),\qquad \tilde{J}\equiv\left(J/T\right)\delta^{-\nu_\perp(1+\zeta)},\qquad \tilde{E}_{\pm}=J\tilde{\rho}_{\pm} \tag{3.13}$$

The critical exponents ν and z are uniquely determined for a given universality class of phase transitions. Although some experiments[23, 24] have claimed to observe a magnetic field dependence in ν and z, such field dependence for all fields contradicts the assumption of second-order phase transitions which asserts universal critical exponents for a given class of transitions. Only by changing the nature of the phase transition as is

expected in the presence of columnar defects can the critical exponents change. Scaling a single set of isotherms as shown in Figure 3.4 is insufficient evidence to verify that the critical scaling is valid, because it is the existence of universal critical exponents that provides evidence for a second-order phase transition. The exponents ν and z must be independent of measurement technique and independent of magnetic field, as long as variations in the vortex separation $\left(\propto 1/\sqrt{H}\right)$ do not change the underlying mechanism of the phase transition. Therefore, for different magnetic fields, the only variable that can be adjusted is the transition temperature $T_M(H)$. Additionally, the functional forms of $\tilde{\rho}_{\pm}(\tilde{f})$ and $\tilde{\chi}_{\pm}(\tilde{f})$ are truly universal for all fields except for a non-universal field dependent coefficient. Furthermore, measurements on different samples with similar vortex-pin interactions must yield the same set of critical exponents. Only by verifying the universality can we provide strong evidence for the phase transition. It is also important to keep in mind that the critical scaling of ρ and χ is only valid for temperatures sufficiently close to T_M. The temperature window is governed by the Ginzburg criterion.[25] Any report of scaling over a temperature range much larger than a value allowed by the Ginzburg criterion[24] is a violation of critical scaling and therefore should not be taken as evidence of a second-order phase transition.

3.4 Finite Size Limitations

As T approaches $T_M(H)$, the growth of ξ_V will eventually be limited by the finite length scale ℓ of the sample as shown in Figure 3.7, and the critical scaling analysis will be expected to break down. In the case of twinned single crystals, the sample length scale ℓ is found to be the average twin boundary separation.[10] Deviation from the scaling behavior is expected to occur within a narrow temperature range $T_M - \Delta T < T < T_M + \Delta T$ for which $\xi_V(T,H) > \sim \ell$, where ΔT is defined by $\ell = \xi_V(T,H) = \xi_0 \left|\Delta T/T_M\right|^{-\nu}$, or

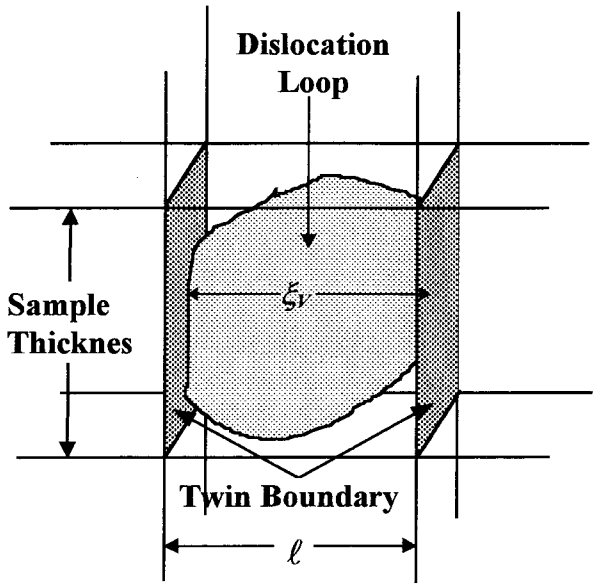


Figure 3.7. Finite size of the twin boundaries limits the growth of the dislocation loop as $T \rightarrow T_M$.

$\Delta T = T_M(\xi_0/\ell)^{1/\nu}$. Using values of ξ_0=400Å, ℓ=2μm, $T_M(H$=10 kOe)=90K and $\nu = \frac{2}{3}$ found previously,[10] we find $\Delta T \approx 0.25$K. This ΔT corresponds to the growth of dislocation loops in the absence of any applied current and represents the maximum ΔT, as can be seen by considering the effects of applying a finite current.

From Eq. (3.2), we see that the thermally determined length scale ξ_V corresponds to a characteristic current J_T, but we can also interpret Eq (3.2) as having a fixed current J correspond to a characteristic probing length ℓ_J given by $\ell_J = \sqrt{\frac{k_B T}{J\Phi_0}}$. Thus, only dislocation loops of size ℓ_J or larger are probed by the current. By applying a higher current, we can investigate dislocation loops which are smaller than the characteristic size ξ_V. We can therefore investigate temperatures closer to T_M than are allowed by the restriction of $\xi_V < \ell$, provided that we apply a current high enough to ensure $\ell_J < \ell$. In other words, we must keep $J > J_\ell$ where J_ℓ is given by

$$J_\ell(T_M) = (k_B T_M)/(\ell^2 \Phi_0). \tag{3.14}$$

We note that this expression for J_ℓ is only valid for $T=T_M$, because we have not consistently accounted for the current dependence correlation length. More accurately, we should define a modified correlation length $\xi'_V(T,H,J)$ which includes the effects of the applied current J.[10]

$$\xi_V'(T,H,J) = \xi_0(H)\left|1-\frac{T}{T_M}\right|^{-\nu} f(x), \qquad x \equiv \frac{J}{J_T(T,H)}. \tag{3.15}$$

The form of the scaling function $f(x)$ is unknown, but we do know that $f(x)\to 1$ for $x\to 0$ and $f(x)\to x^{-1/(d-1)}$ for $x>>1$.[10] For temperatures $T \to T_M$, any non-zero current will give $x>>1$ so that $\xi_V'(T \to T_M) = \left[k_B T_M / J\Phi_0\right]^{1/(d-1)}$. Critical scaling behavior then breaks down when $\xi_V' = \ell$ which corresponds to the result obtained in Eq. (3.14). Previous dc transport experiments confirm that critical scaling breaks down near T_M for $J<J_\ell \approx 2 \times 10^5$ Am^{-2}, which corresponds to $\ell \approx 2\mu m$ in good agreement with the twin boundary separation.[10] Imposing a lower bound on the applied dc current poses a significant problem in interpreting the scaling results of dc current-voltage measurements. Further complications arise as we consider current-voltage characteristics taken at high current densities because vortex dissipation becomes dominated by flux flow which follows ohmic behavior rather than critical scaling behavior.[26] Additionally, applying high currents leads to the problem of joule heating due to non-zero contact resistance. With limiting conditions on both the low current and high current, it is difficult to perform scaling analysis over a broad range of currents, and it has been shown that with only a limited current range, the critical scaling of current-voltage characteristics near a phase transition cannot be distinguished from the predicted behavior for flux creep.[27, 28] This limitation in performing scaling analysis on dc current-voltage measurements raises questions about the validity of using such analysis to provide conclusive evidence for a second-order vortex phase transition. It is therefore essential to develop ac measurements that analyze resistivity versus frequency without being subjected to the same limitations.

By applying an ac current at frequency f, the frequency introduces a new length scale that reduces the constraint imposed by the finite size effect. In the low current limit,

the finite size limitation in the frequency occurs when $\xi_V(T,H,f_\ell)=\ell$ where again ℓ is expected to be approximately the twin boundary separation. Unfortunately, the scaling relation for frequency, $f_T \sim \xi_V^{-z}$, is asserted by the theory of critical phenomena. Unless the underlying critical dynamics is understood, the relation $f_T \sim \xi_V^{-z}$ does not provide a clear physical interpretation as is the case for the current scaling relation in Eq. (3.2). Although we can see that $f_\ell \sim \ell^{-z}$, the proportionality coefficient in this relation is not known. It is therefore difficult to estimate the lower bound on the frequency due to the finite size effect. The measurements of ac impedance performed in Chapter 5 confirm that consistent scaling behavior can be shown with f>~10^4 Hz for temperatures $T \approx T_M$, which places an empirical limit of the finite size effect of f_ℓ<10^4 Hz.

Another concern in performing ac measurements is the amplitude of the applied current density. As shown in Figure 3.3, the characteristic current density J_T separates the current dominated regime from the thermal dominated regime. As long as the excitation current density J satisfies $J << J_T(T,H)$, the vortex response is dominated by thermal fluctuations and the effects of the finite current density can be ignored. It is important to be in the thermal dominated regime when performing the frequency dependent measurements of $\rho(f)$ and $\chi(f)$, because in a regime where both current and frequency effects are important, both quantities enter the scaling relation together. However, the techniques for performing scaling with two variables have not been established theoretically. We note that in the case of the ac impedance measurements described in Chapter 5, the applied current density is J=2×10^4Am^{-2}. From Figure 3.3, this current will only be in the current dominated regime for temperatures within the interval $|T - T_M|$ <~ 0.006K for an applied field of H=1kOe. At higher magnetic fields, the current dominated regime is broadened slightly, so that for H=50 kOe the current effects become dominant for temperatures

within the interval $|T - T_M| <\sim 0.04\,\mathrm{K}$. These temperature intervals for all fields of our investigation are roughly equal to or smaller than the temperature steps taken in the experiments described in Chapter 5. Therefore, all ac data are in the thermal dominated regime with the possible exception of one or two temperatures closest to T_M, which justifies the use of the frequency scaling analysis in Eqs. (3.3)-(3.6).

This chapter has derived various critical scaling relations and experimental signatures associated with a second-order vortex-solid to vortex-liquid phase transition. These scaling relations will be used to analyze the experimental results obtained in Chapters 5 through 8. One of the most important points in using the scaling relations is that the critical exponents ν and z must be universal, independent of magnetic field range, applied current, applied field, and experimental technique. These exponents define the nature of the phase transition which can only be changed with fundamental changes to the experimental system such as the introduction of columnar defects. Only by verifying the universality of the exponents over a range of magnetic fields and experimental techniques can the existence of a second-order vortex-solid to vortex-liquid phase transition be verified. Furthermore, the limitations of applying the critical scaling analysis to experimental results, such as finite size effects and finite current density in ac experiments, must also be considered in order to ensure the validity of the analysis.

References

[1] D. S. Fisher, M. P. Fisher, and D. A. Huse, Phys. Rev. B **43**, 130 (1991).

[2] A. T. Dorsey, Phys. Rev B **43**, 7575 (1991).

[3] D. R. Nelson and V. M. Vinokur, Phys. Rev. Lett. **68**, 2398 (1992); Phys. Rev. B **48**, 13060 (1993).

[4] D. S. Reed, N.-C. Yeh, M. Konczykowski, and A. V. Samoilov, submitted to Phys. Rev. Lett. (1994).

[5] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, M. Konczykowski, and F. Holtzberg, submitted to Phys. Rev. Lett. (1994).

[6] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, D. A. Beam, and F. Holtzberg, Phys. Rev. B **49**, 4384 (1994).

[7] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6150 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 413.

[8] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, F. Holtzberg, M. Konczykowski, C. C. Tsuei, and C. C. Chi, Physica A **200**, 374 (1993); to appear in Physica C, (1994).

[9] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, T. A. Tombrello, A. P. Rice, and F. Holtzberg, Phys. Rev. B **47**, 8308 (1993).

[10] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6146 (1993).

[11] N.-C. Yeh, D. S. Reed, W. Jiang, U. Kriplani, C. C. Tsuei, C. C. Chi, and F. Holtzberg, Phys. Rev. Lett. **71**, 4043 (1993).

[12] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, D. A. Beam, M. Konczykowski, T. A. Tombrello, and F. Holtzberg, Phys. Rev. Lett. **72,** 550 (1994).

[13] N.-C. Yeh, W. Jiang, D. S. Reed, A. Gupta, F. Holtzberg, and A. Kussmaul, Phys. Rev. B **45**, 5710 (1992).

[14] N.-C. Yeh, D. S. Reed, W. Jiang, U. Kriplani, F. Holtzberg, A. Gupta, B. D. Hunt, R. P. Vasquez, M. C. Foote, and L. Bajuk, Phys. Rev. B **45**, 5654 (1992).

[15] See, for example, S. K. Ma, *Modern Theory of Critical Phenomena* (Addison-Wesley, Reading, MA, 1976).

[16] E. H. Brandt, Physica C **162**, 1167 (1989)

[17] S. A. Wolf, D. U. Gubser, and Y. Imry, Phys. Rev. Lett. **43**, 324 (1979).

[18] M. E. Fisher, M. N. Barber, and D. Jasnow, Phys. Rev. A **8**, 1111 (1973).

[19] M. Konczykowski, F. Rullieralbenque, E. R. Yacoby, A. Shaulov, Y. Yeshurun, and P. Lejay, Phys Rev B **44,** 7167 (1991).

[20] L. Civale, A. D. Marwick, T. K. Worthington, M. A. Kirk, J. R. Thompson, L. Krusin-Elbaum, Y. Sun, J. R. Clem, and F. Holtzberg, Phys. Rev. Lett. **67**, 648 (1991).

[21] W. Gerhauser, G. Ries, H. W. Neumuller, W. Schmidt, O. Eibl, G. Saemannischenko, and S. Klaumunzer, Phys. Rev. Lett. **68**, 879 (1992).

[22] T. Hwa, P. Ledoussal, D. R. Nelson, and V. M. Vinokur, Phys Rev L **71,** 3545 (1993).

[23] Y. Ando, H. Kubota, and S. Tanaka, Phys. Rev. Lett. **69**, 2851 (1992).

[24] J. M. Roberts, B. Brown, B. A. Herman, and J. Tate, Phys. Rev. B **49**, 6890 (1994).

[25] U. Welp, S. Fleshler, W. K. Kwok, R. A. Klemm, V. M. Vinokur, J. Downey, B. Veal, and G. W. Crabtree, Phys. Rev. Lett. **67,** 3180 (1991).

[26] M. N. Kunchur, D. K. Christen, and J. M. Phillips, Phys. Rev. Lett. **70**, 998 (1993).

[27] E. Zeldov, N. M. Amer, G. Koren, A. Gupta, R. J. Gambino, and M. W. McElfresh, Phys. Rev. Lett. **62,** 3093 (1989).

[28] S. N. Coppersmith, M. Inui, and P. B. Littlewood, Phys. Rev. Lett. **64,** 2585 (1990).

Chapter 4 : General Experimental Procedures

Before considering the experimental techniques specific to high frequency ac measurements, there are many procedures common to all measurements in high temperature superconductivity. These techniques include sample preparation, electrical contacts, and temperature control. Although these materials are called high temperature superconductors, the temperature range of interest is still quite low (70K~95K for $YBa_2Cu_3O_7$, depending on the applied magnetic field) and measurements must be performed in cryogenic dewars. The experiments also require accurate temperature control to better than ±0.01K, and high magnetic fields requiring a superconducting magnet. This chapter discusses the preparation of the superconducting samples

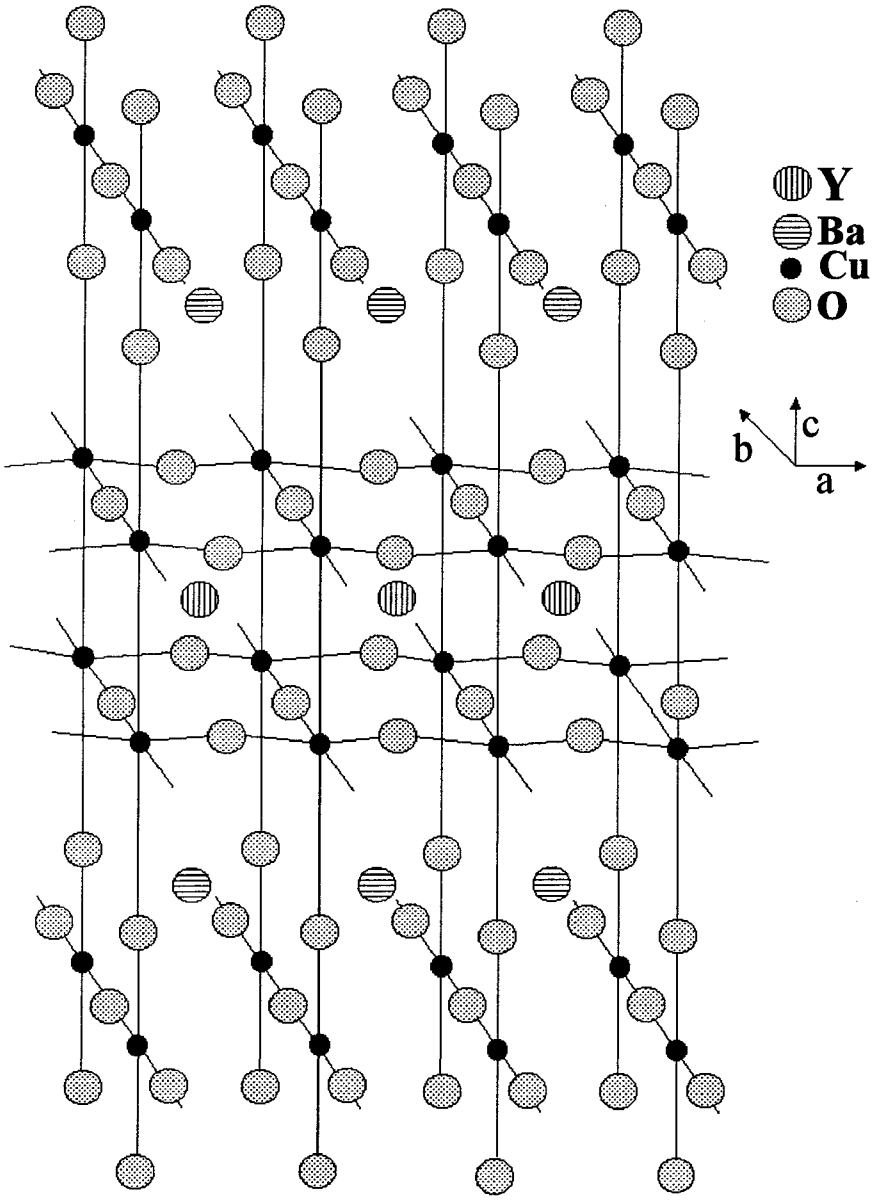


Figure 4.1. Diagram of the atomic structure of $YBa_2Cu_3O_7$[1].

measured in this work as well as the cryogenic dewars and sample probes used in the measurements. Details of each of the experimental techniques and electronic instrumentation used in performing measurements over a broad frequency range are included in Chapters 5-7 along with the data obtained from each of the techniques.

4.1 High Temperature Superconducting Materials

The high temperature superconductor used throughout this work is $YBa_2Cu_3O_7$ which has a complex atomic structure shown in Figure 4.1. The complex atomic structure is common to all of the high temperature superconductors, but more interestingly all of the high temperature superconductors have the same copper oxide (CuO_2) plane. Although some high temperature superconductors have more layers of copper oxide per unit cell, the copper oxide plane remains a quasi two-dimensional structure within the "perovskite" material and it has been pointed to as the essential feature responsible for the high transition temperatures found in these materials.[2] However, the connection between the quasi two-dimensional structure and the high transition temperature has not been shown conclusively and the precise mechanism of high temperature superconductivity remains unknown. All of the high temperature superconductors are also highly anisotropic in their electronic properties with conduction in the ab plane being much better that the conduction along the c-axis. This sample anisotropy can be expressed in terms of the anisotropy in the effective masses of the carriers,[3] $\varepsilon^{-2} = \frac{m^*_{c-axis}}{m^*_{ab-plane}} \approx 60$ for $YBa_2Cu_3O_7$, where m^*_{c-axis} and $m^*_{ab-plane}$ are the effective mass of carriers along the c-axis and within the ab plane respectively. Actually, $YBa_2Cu_3O_7$ is one of the more isotropic high temperature superconductors, as compared to $Bi_2Sr_2CaCu_2O_8$ which has an anisotropy ratio of[4] $\varepsilon^{-2} \approx 10^4$.

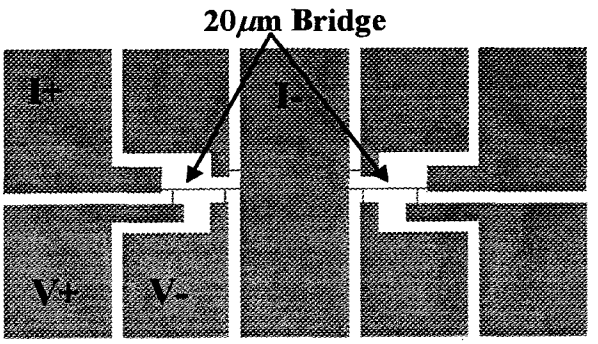


Figure 4.2. Thin film patterned for resistivity measurements. The two bridges are 600μm long × 20μm wide to provide a large resistance signal. The large pads are used for electrical contacts with the current and voltage configuration indicated for the left bridge. The same technique can be used for Hall effect measurements.

These high temperature superconductors are readily available in three forms, single crystals, thin films, and polycrystalline materials. Polycrystalline materials (ceramics) are useful in the fact that they can be made into large bulk samples, but they show much lower critical current densities due to the presence of grain boundaries, $j_c \approx 500\,\mathrm{Acm}^{-2}$.[5] In addition, the network of grain boundaries between crystals makes it difficult to measure the intrinsic properties of the superconducting material. Thin films have proven to have much higher potential for applications because of their high critical current densities, $j_c > 10^7\,\mathrm{Acm}^{-2}$ at 64K and zero applied field, and $j_c > 10^5\,\mathrm{Acm}^{-2}$ in an applied field of 6 Tesla.[6] Another advantage in using films is that they can be patterned using standard lithographic techniques and are currently used in devices such as high temperature SQUIDs[7] and microwave filters.[8] For measurements of fundamental superconducting properties, films are often patterned into a long narrow bridge as shown in Figure 4.2 to give a higher signal to noise ratio in resistivity measurements. Patterning the films also offers the advantage of having large electrical contacts which give a low contact resistance, and maintaining a uniform current distribution in the sample. Figure 4.2 shows a pattern we used for some preliminary measurements on thin films. However, film growth is not a smooth layer by layer growth, but a collection of individual grains which in some cases grow as screw dislocations ~1μm in diameter.[6] These grain boundaries and dislocations can lead to weak links between grains and in

some cases the weak link effects can significantly alter the measured results,[9] though high quality films are unaffected by these grain boundaries.

High quality single crystals are relatively free of defects and provide the cleanest system for experimental measurements. Although there are no grain boundaries to interfere with the measurements, most single crystals contain twin boundaries which separate different regions of the sample in which the a and b axes are rotated by 90°. Typically, the twin boundaries in these samples are ~10μm apart. Single crystals of $YBa_2Cu_3O_7$ typically grow to sizes of only ~1mm×1mm×25μm thick with the smallest dimension along the c-axis. The small size as well as the brittle nature of the crystals makes preparation of the samples considerably more difficult than the preparation of films or ceramic samples. However, single crystals provide the best opportunity to look into the intrinsic properties of $YBa_2Cu_3O_7$ and therefore are chosen for all experiments presented here.

4.2 Sample Preparation

The samples used in this work are $YBa_2Cu_3O_7$ single crystals made by F. Holtzberg.[10] A mixture of YBa_2Cu_3O, $BaCuO_2$, and CuO are pressed into a pellet which is placed in a solid gold crucible and heated to 975°C to melt the pellet. The mixture is cooled slowly at a rate of 5°C per hour to allow the growth of the crystals. After the melt cools, the crystals are carefully removed from the remaining melt. The crystals are found to be oxygen deficient so they are annealed in O_2 for ten days at 420°C[10] to ensure that oxygen fully diffuses into the crystal.

The superconducting properties of the crystal are very sensitive to the presence of any impurities, so that even small traces of many contaminants can severely degrade the sample quality. Standard crucibles such as those made of MgO are not suitable for the growth because a small amount of the crucible combines with the melt and can contami-

nate the crystals. Although the gold crucible that is used also "contaminates" the melt, the presence of a small amount of gold actually enhances the T_c of the crystals. A small number of gold atoms displace the copper atoms giving an atomic structure $YBa_2Cu_{2.98}Au_{0.02}O_{7-x}$ ($x=0.03\pm0.02$). The presence of gold atoms appears to improve the T_c of the samples by increasing the oxygen content by ≈ 0.015[11] for each one atomic percent of Au^{3+}. In these crystals, the gold concentration of 0.02 contributes an estimated 0.03 to the oxygen content, which can enhance T_c by as much as 7K.[12]

The $YBa_2Cu_3O_7$ crystals are moderately air sensitive,[13,14] and after extended exposure to air they form a non-stoichiometric surface layer that must be removed before good electrical contacts can be made for transport measurements. A chemical etch consisting of a 1% solution of Br_2 in ethanol which preferentially removes the non-stoichiometric surface layer was developed by R. Vasquez at the Jet Propulsion Laboratory[15] and is used in our sample preparation. A single crystal requires approximately 10 minutes in the Br_2 etch to fully remove the bad surface layer, and after the etch, the sample is rinsed twice in separate beakers of pure ethanol. The improved surface quality can be seen from a sharpening of the magnetic transition as well as the presence of a clear Fermi edge in x-ray photoelectron spectroscopy (XPS) studies[15] which are sensitive to the sample surface quality. It should be emphasized that the Fermi edge of $YBa_2Cu_3O_7$ crystals cannot be seen before the Br_2 etch.

In order to make electrical contacts, the sample must be masked in preparation of evaporating gold onto the corners and etched in the manner described above. Because the sample surface degrades in air after 5~10 minutes, the sample must be masked before the etch is begun rather than after it is completed. As shown in Figure 4.3, the sample is masked with gold foil in such a way as to leave the corners exposed for evaporation of gold pads, and the ends of the foil are attached to a glass slide with silver paint. Gold foil

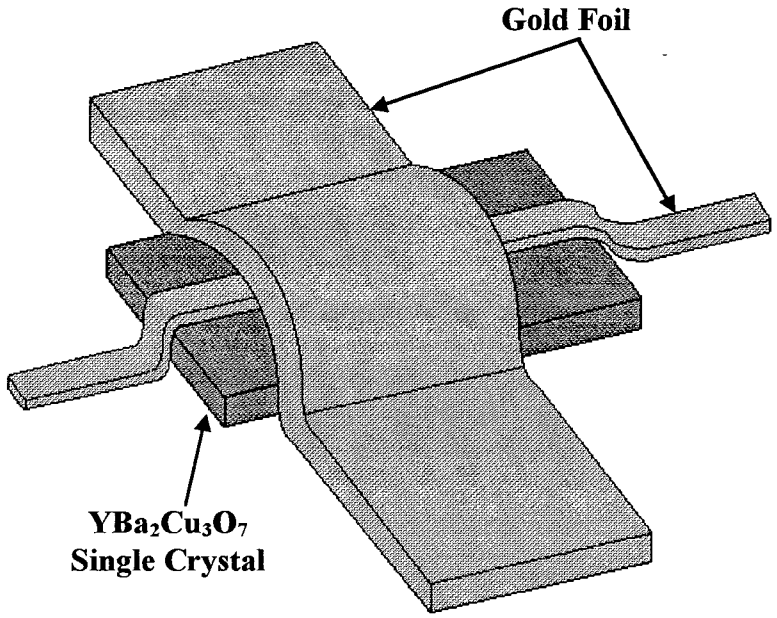


Figure 4.3. $YBa_2Cu_3O_7$ crystal masked with 0.002" Au foil in preparation of Br_2 etch and deposition of Au contact pads.

and silver paint are both found to be able to withstand the Br_2 etch for the time needed, though after several hours in the Br_2 etch, a large portion of the gold foil will dissolve. It is also important to give the silver paint sufficient time to dry (usually overnight is enough) or it will not hold the mask during the etch. Immediately after the etch and rinse is completed, the sample is loaded into a vacuum chamber and the chamber is pumped down to $<10^{-8}$ torr. Approximately 2500Å of gold is then evaporated onto the exposed corners of the sample. Following the evaporation, the sample is annealed at 300°C in O_2 for 24 hours. After annealing, the samples have a contact resistance of ~1Ω at room temperature, and slightly higher at temperatures of 75K to 95K where most of the experiments are performed.

After the contact pads are made on the crystal, it is mounted onto a 0.375 inch diameter sapphire disk using a small amount of thermally conductive grease (Appezone N grease) and electrical leads are attached from the sample to the sapphire substrate. Because most crystals grow with the c-axis direction much thinner (~25μm) than the a or b directions (<~1000μm), the crystals are mounted so that the c-axis is perpendicular to the surface of the substrate. Two methods have been developed to attach leads between the gold pads on the crystal and the substrate. The first method, shown in Figure 4.4(a), uses

0.001 inch diameter gold wire attached with an ultrasonic wire bonder from the sample to gold pads evaporated onto the sapphire. Wire bonding was used for many of the initial experiments, but the vibration of the wire bonder tends to degrade the contact between the sample and the gold contact pad. The extremely brittle nature of the $YBa_2Cu_3O_7$ crystals also makes using a wire bonder dangerous as the sample is easily broken if the bonding tip

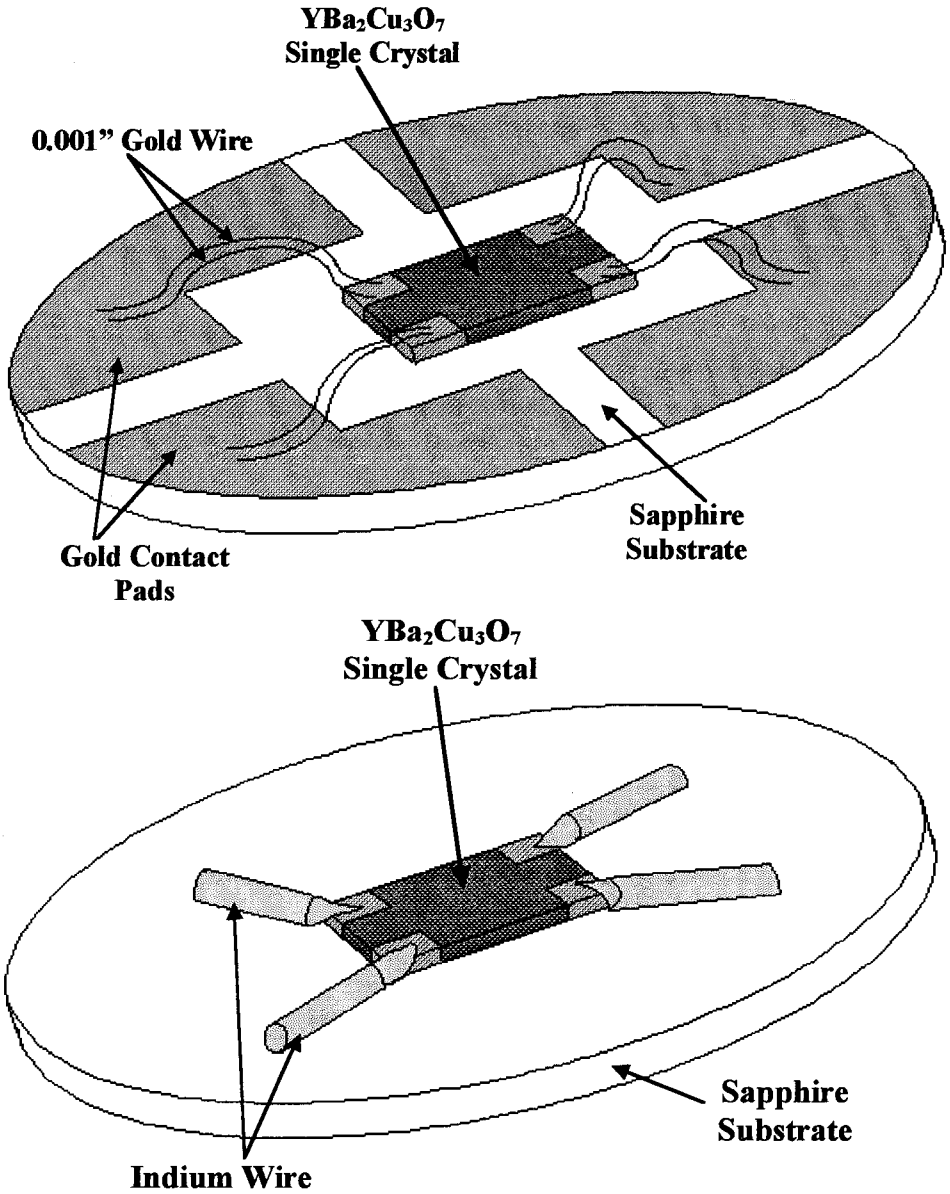


Figure 4.4. $YBa_2Cu_3O_7$ crystal after mounting it onto the sapphire substrate. Electrical contacts are made with either **(a)** 0.001 in. gold wire using an ultrasonic bonder or **(b)** indium pressed onto the contact pads.

is allowed to apply too much force to the crystal. However, the bond is very weak with such a small force and the bond is easily broken. Wire bonding also has the disadvantage that the substrate must also have gold contact pads so the other end of the wire can be bonded to the substrate. This substrate preparation is a lengthy procedure because gold does not stick to sapphire directly so a buffer layer of aluminum must be evaporated onto the sapphire to provide adhesion for the gold. An alternative method shown in Figure 4.4(b) uses 0.005 inch diameter indium wire pressed into place. The indium wire is laid out on the sapphire substrate before the sample is put in place and the indium is pressed down to make it stick to the sapphire. The sample is then positioned with the gold pad near the end of the indium wire, and the indium is gently spread out onto the sample pad. The indium is malleable enough that it can easily be spread using a sharp plastic tool. Although pressing the indium requires a very steady hand, the bonds have proven to be less likely to break and generally show a lower contact resistance.

4.3 Experimental Environment — Cryogenic Dewars

The experiments to be performed require a well defined sample environment: temperature stabilized to better than ±0.005K in the range $T \approx 75\text{K} \rightarrow 95\text{K}$, and a high dc magnetic field up to H~9 Tesla. This temperature range requires a cryogenic dewar for performing experiments, and the high magnetic field requires the use of a superconducting magnet shown in Figure 4.5(b). Some measurements require lower fields, H~0.1 Tesla, below the residual field of some superconducting magnets, so a copper coil magnet is used as shown in Figure 4.5(a) for the low field experiments H<0.6 Tesla. Using the copper coil magnet also greatly reduces operating costs by eliminating the liquid helium (LHe) required for the superconducting magnet.

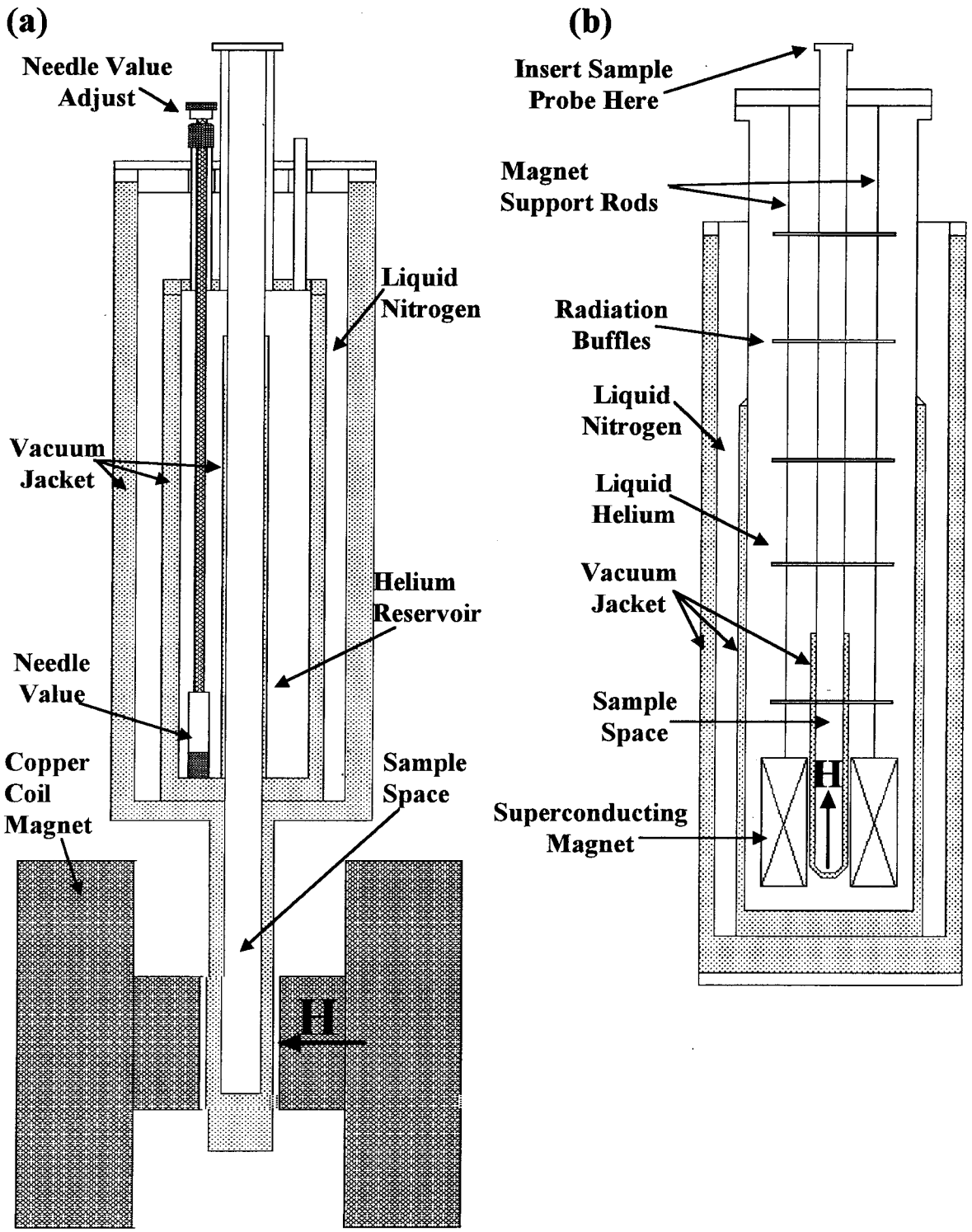


Figure 4.5. Cryogenic dewars (not to scale) used for performing experiments with **(a)** Copper Coil magnet for fields <6kOe, and **(b)** a superconducting magnet for fields up to 90 kOe.

Using the low field dewar shown in Figure 4.5(a), the sample is placed near the base of the narrow "tail" which is inserted into the gap of a standard copper coil magnet. The magnet provides a horizontal magnetic field up to 6kOe. Although the temperature is controlled with the use of a heater and temperature sensor located on the sample probe (see section 4.4), the cooling power must be provided by the dewar. The dewar is equipped with a "super varitemp" temperature control system built by Janis Research Co. and is shown in Figure 4.6. This design incorporates a small capillary tube connecting the liquid helium reservoir to the sample area. The flow of LHe (and therefore the cooling power) is controlled from outside the dewar by adjusting the needle valve, and a small heater is located in the base of the sample area to vaporize the LHe and increase the cooling of the sample. However, the needle valve cannot control the flow of LHe accurately over a long period of time, and constant adjustments to the needle valve are required to keep the flow constant. Additionally, the He vapor cooling the sample is at a temperature near 4.2K while the sample is typically held near 90K, so there is a large temperature gradient between the sample and the base of the

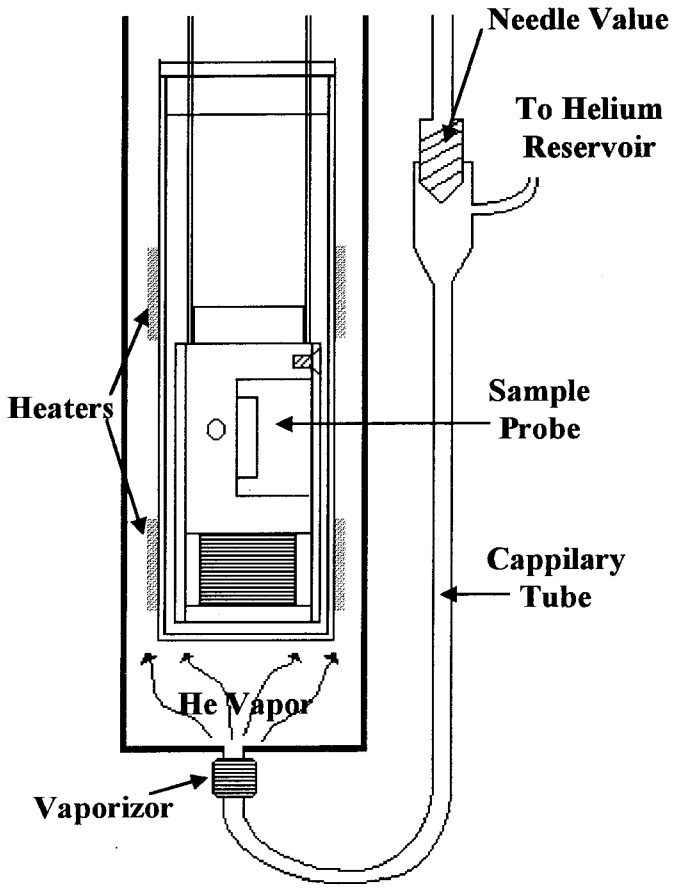


Figure 4.6. The "super varitemp" temperature control system uses a small flow of cold helium vapor to provide cooling for the sample. The needle valve controls the flow rate of the helium, thereby controlling the cooling power provided to the sample.

sample probe. In order to improve the reliability of the temperature control, it is found that liquid helium is not necessary for temperatures T>~80K. The liquid nitrogen in the surrounding nitrogen reservoir cools the He gas in the helium reservoir to a temperature of ~77K. By applying a steady flow of He gas into the helium reservoir, the cold He gas will flow through the capillary tube providing enough cooling to control the sample temperature. With the needle valve left open, the flow rate can be accurately controlled with an external regulator to a rate of ≈0.5 *l*/min. Using He gas, the temperature gradient within the sample space is greatly reduced and the steady gas flow keeps the cooling power constant over long periods of time resulting in improved temperature stability of ±0.003K and more reliable operation.

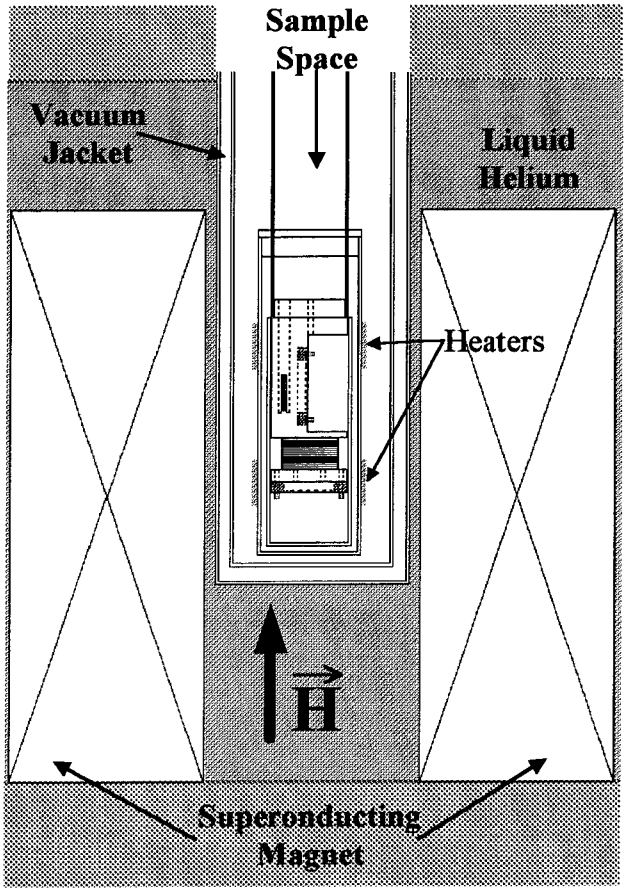


Figure 4.7. Detail of the superconducting magnet sample space. The magnet is submersed in liquid helium. The vacuum jacket and low pressure of He gas in the sample space provide a weak thermal link to cool the sample.

The superconducting magnet system shown in Figure 4.5(b) is quite different from the low field system in several important ways. The superconducting magnet is a single solenoid placed in the liquid He with the sample inserted into the center bore. Because the superconducting magnet must remain in a LHe bath, the sample is cooled through a weak thermal link to the LHe as shown in Figure 4.7. The sample space is enclosed by a stainless steel tube which extends to the top of the dewar. The vacuum jacket shown in Figure 4.7 extends ≈40 cm above the sample to prevent

direct thermal contact between the LHe bath and the sample. Approximately 0.1mbar of He gas (as measured when the entire dewar is at room temperature) is inserted into the sample space to provide a limited heat exchange for controlling the temperature. As with the low field dewar, the temperature is controlled with a heater and temperature sensor located on the sample probe.

4.4 The Sample Probe

The sample probe provides the link between the sample to be measured and the external instrumentation. It must provide for a rigid physical support, temperature control, and allow for the electrical connections and sample mobility required for the experiments. The experiments to be performed require a temperature stability better than ± 0.005K, and ac measurements over a frequency range f=100Hz $\rightarrow$ 3MHz with a noise level of $\sim 10^{-8}$V. The sample probes must also be vacuum sealed to allow the air and any moisture to be pumped out before the dewar is cooled and to prevent any additional air from leaking in and condensing and freezing inside the probe. A slow leak is less critical in the low field dewar because it operates with the pressure of He gas slightly above atmospheric pressure inside the sample space, but the vacuum seals in the high field dewar are critical because it operates at a low pressure of He gas ($\sim 10^{-4}$ torr). If the probe leaks in the high field dewar, air will continually flow into the sample area and then condense and freeze, making temperature control unreliable. Furthermore, if too much air leaks into the sample area, the accumulated ice and gas pressure can damage the sample and probe.

Although there are slight variations among sample probes, Figure 4.8 shows the general design used for the top section of the probe that extends outside the dewar. Flexible coaxial cables are used for measuring the sample voltage signals to give the required noise level and frequency range, but care must be taken in designing and construct-

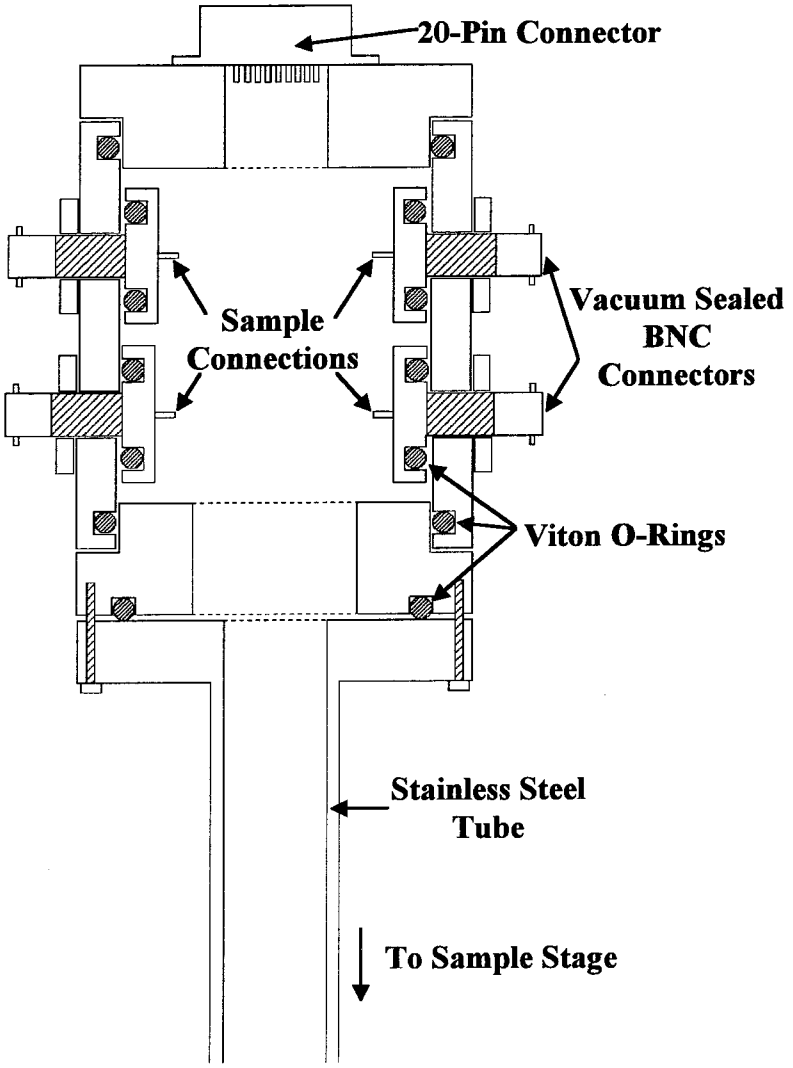


Figure 4.8. Top section of the sample probes showing the vacuum tight connections for coaxial wiring and the multi-pin connector for the heater and temperature sensors. The stainless steel tube runs down the length of the dewar and holds the sample stage in place.

ing the probe to maintain proper shielding of the cables. The vacuum sealed BNC feedthroughs shown in Figure 4.8 permit a complete coaxial path through the vacuum seal of the probe. Additional connections for the sample heater and temperature sensor are provided by a single 20-pin connector at the top of the probe. The stainless steel tube runs down the length of the probe to hold the sample stage in place.

4.5 AC Transport Probe

The sample stage shown in Figure 4.9 is designed for performing ac impedance measurements. The sample stage is machined from a copper block and has two positions for samples to match the two different magnetic field orientations in the high field dewar and low field dewar (see Figure 4.5 and Figure 4.7). Most measurements are made with the sample c-axis parallel to the dc magnetic field H. For the low field dewar, aligning

H||c-axis requires mounting the sample with the c-axis horizontal which corresponds to the primary sample position in Figure 4.9. For the high field dewar, the magnetic field is vertical so the sample must be mounted with the c-axis vertical, or in the secondary sample position. Alternately, if measurements are to be made with the field H in the crystal ab plane then the high-field measurements are made with the sample in the primary position, but for low field measurements the sample remains in the primary position and the probe is rotated by 90° about its axis.

A carbon-glass temperature sensor is placed directly behind the primary sample position, and a capacitance sensor is placed directly under the secondary sample position. The sample stage is covered with a copper shield to provide a more uniform temperature for the space directly surrounding the sample, and the copper shield is covered by a second can to further shield the sample stage from the cold He vapor. Coarse temperature control for the region surrounding the sample probe can be maintained with the heaters placed on the outer shield as shown in Figure 4.6. An additional 50Ω heater foil is placed directly on the sample stage between the two sample positions and is connected to a LakeShore 93C temperature controller for fine temperature control. By placing the heater below the sample and sensor, the primary temperature gradient is below the heater so that the temperature gradient between the

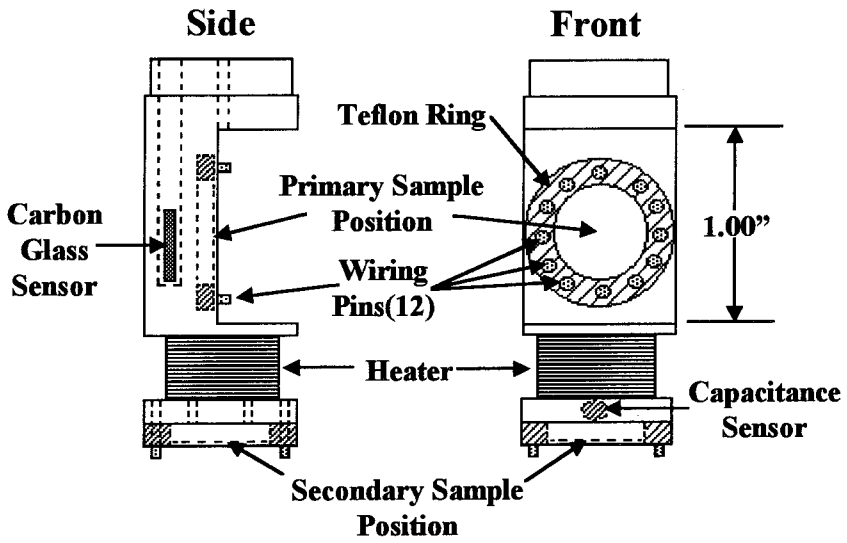


Figure 4.9. Sample stage for ac transport measurements.

sample and carbon glass sensor is minimized. For high field measurements, the capacitance sensor shows little field dependence in its temperature reading,[16] so it is used for controlling the temperature near the secondary sample position. Temperature gradients are not found to be a serious problem in the high field dewar because the cooling is not directed from the bottom as it is in the low field dewar. With this probe, temperature stability of ±3mK can routinely be achieved, with the stability limited only by the resolution of the LakeShore temperature controller. In some recent experiments, temperature resolution of ±1mK was achieved by using a computer to control the heater power directly rather than using a commercial temperature controller.[17]

The coaxial cables are soldered to copper pins that surround the sample position. These pins are screwed into a ring of Teflon to provide electrical isolation between pins and to the sample stage. Electrical contacts are then made to the sapphire sample substrate with fine copper wires that are also soldered to these pins and held onto the substrate with pressed indium. The connections between the substrate and sample are described in section 4.2. By running the coaxial cables to the pins directly surrounding the substrate, the coaxial shielding is maintained as close to the sample as possible, leaving only a short length (<1cm) of unshielded wire.

4.6 Rotatable AC Transport Probe

In order to investigate the anisotropic properties of the samples, it is necessary to be able to rotate the sample to various angles θ with respect to the applied field. We use the notation θ to be the angle between the applied field H and the crystal c-axis. Rotating the sample in the low field dewar is relatively easy because the field is horizontal (see Figure 4.5(a)). Rotating the sample in a horizontal magnetic field is accomplished by just rotating the entire sample probe about its vertical axis. However, with a standard center-

bore solenoid superconducting magnet shown in Figure 4.5(b), the field is aligned vertically so rotating the entire probe along its vertical axis does not change the angle θ. For rotation in the superconducting magnet, the sample stage itself must be designed in such a way as to allow rotation along an axis perpendicular to the magnetic field. This rotation is accomplished by making the sample stage in the form of a small copper ball as shown in

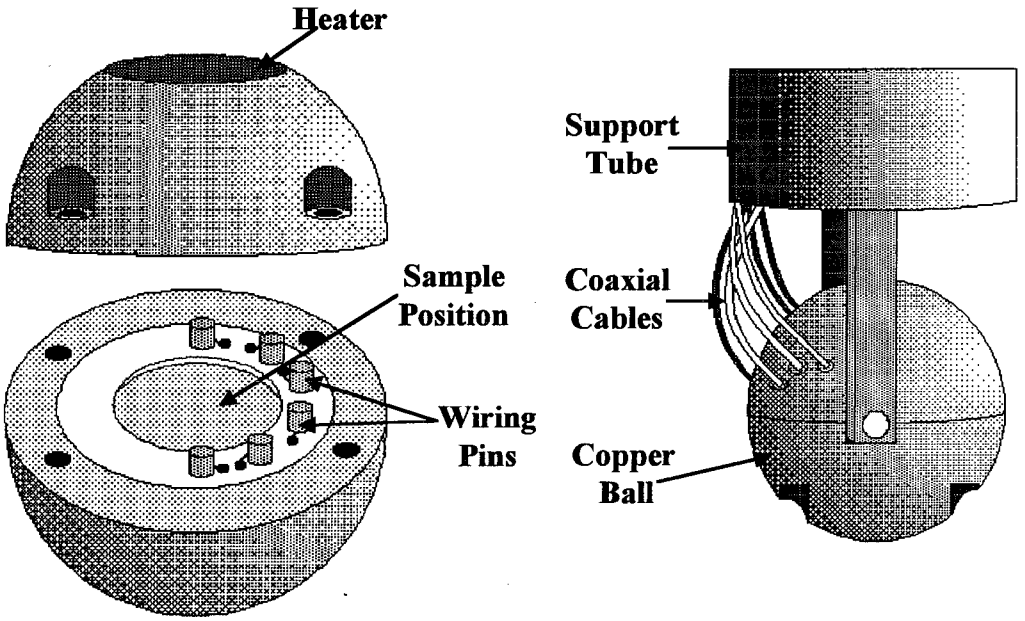


Figure 4.10. Ball Sample stage that permits rotation of the sample in the high field dewar. This drawing is a preliminary sketch used to plan the actual construction which is shown in Figure 4.11.

Figure 4.10. This copper ball is rotated along a horizontal axis by the stainless steel rod attached to the side of the probe as indicated in Figure 4.11. The angle of the sample is selected by adjusting the height of the steel rod, and the height of the rod is adjusted with the threaded screw at the top of the probe.

Temperature is controlled in the same manner as the sample stage described in section 4.5. The copper ball provides an environment of uniform temperature surrounding the sample. The copper ball is surrounded by an outer can which has heaters to provide coarse temperature control of the local environment surrounding the sample stage. A

temperature sensor is placed inside the copper ball directly under the sample and a heater is attached to the ball. A LakeShore temperature controller uses the sensor and heater to maintain a temperature control of ±0.003K. Electrical contacts are also made in a manner similar to that described in section 4.5. The coaxial cables are soldered to pins surrounding the sample as shown in Figure 4.10, and small copper wires are used to connect these pins to the sample substrate. The wiring pins are anchored in a Stycast epoxy ring that provides electrical isolation, but good thermal contact to the copper ball to provide a good heat sink for the coaxial cables.

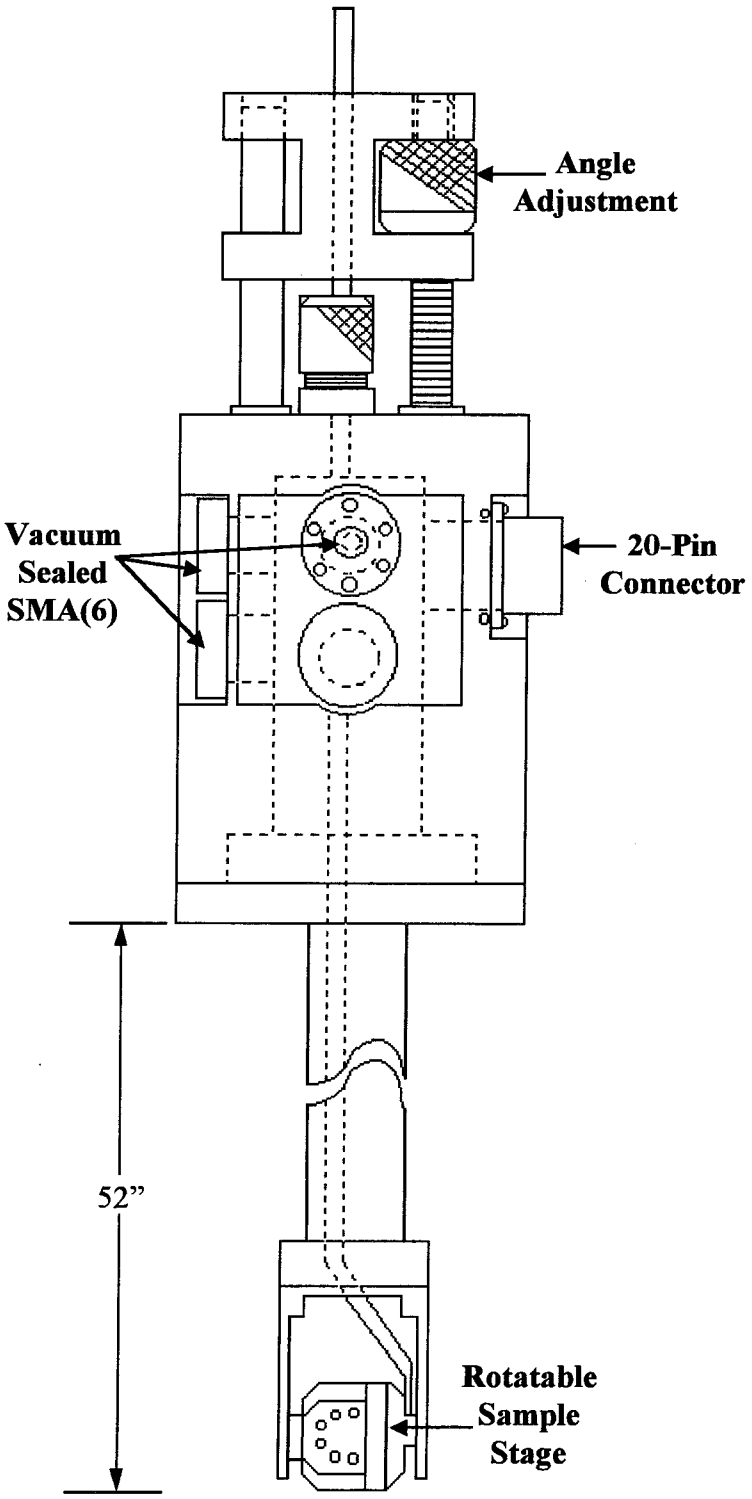


Figure 4.11. Drawing of the rotational mechanism for the ball probe as it was actually constructed.

4.7 AC Susceptibility Probe

Another probe used for a large portion of this work is the ac magnetic susceptibility probe. The basic design is shown in Figure 4.12 and is quite similar to the sample stage used for ac transport measurements described in section 4.5. The temperature sensors and heater are essentially identical to those of the transport probe, with the only difference being the use of two small copper coils which are used to make the susceptibility measurements. The coils used have a 0.125 inch inner diameter made from 24 turns of 0.002 inch copper wire and are mounted on a sapphire substrate with GE varnish. The leads from the coil are connected to pins on a Teflon bracket which also acts to hold the sapphire substrate in place. Coaxial cables are also soldered to the pins and run to the top of the probe. Because of the small size of the coil, both the primary and secondary sample positions are located within the central section of the sample stage, which has the advantage of keeping both coils at nearly identical temperatures. Keeping the coils at the same temperature is important because the empty coil is used as a reference coil in the susceptibility measurements. More details of the coils and how they are used for susceptibility measurements are given in Chapter 6.

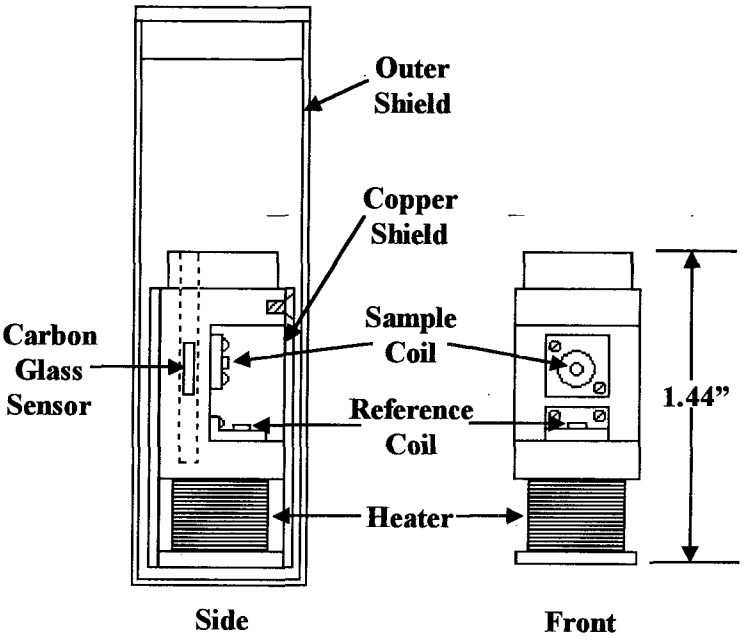


Figure 4.12. Susceptibility Sample stage for probe used to make magnetic susceptibility measurements.

4.8 Computer Control

One set of data for a single value and angle of magnetic field such as that shown in Figure 5.6 involves over 4×10^4 individual impedance measurements. In addition, the temperature is monitored throughout the scan of each isotherm to ensure good temperature stability, the ac current amplitude and frequency must be set, liquid nitrogen must be filled into the dewar, and the magnetic field must be set to the correct value. With all of the electronic instruments being used together, and the quantity of data that must be gathered for analysis, it is essential to have reliable computer control of the experiment. Computer control involves two aspects: the hardware of the interface and the control program used by the computer. A standard IEEE-488 interface is available on most electronic instruments and provides a flexible high speed interface that works well for most applications. With all of the computer power available at reasonable prices, elaborate commercial software packages such as Labview® have become popular methods of developing computer control routines. However, when this thesis work was begun in 1989 the cost of such a system was beyond the reach of a typical laboratory budget and it was common practice, especially for experienced programmers, to start from scratch and write a control program in a standard language such as C.

4.8.1 Interfacing the Computer to the Instruments

The interface used is a standard IEEE-488 parallel interface that has become quite popular and is available on many electronic instruments. In this IEEE protocol, all devices are connected to the same data bus line, and a series of address lines identify which device is to read or write data over the data lines[18]. The address lines and read/write lines are controlled by a board installed in the computer. Communications over the bus take the form of a string of ASCII characters transmitted 1 byte at a time. Interpretation of the

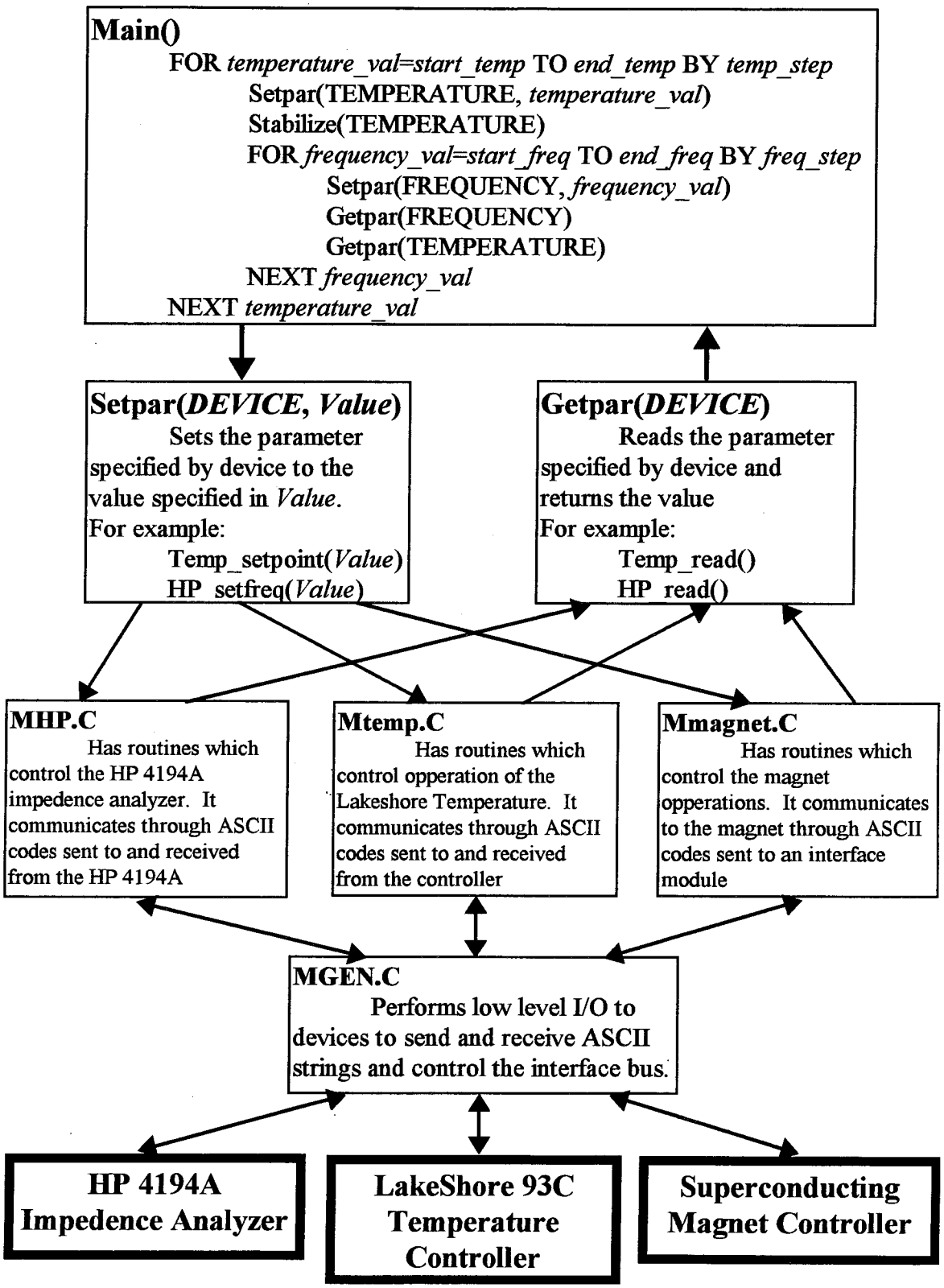


Figure 4.13. Block diagram of the control program Mscan showing a simplified scan routine which accesses only the three devices shown at the bottom.

ASCII string and the command functions contained in the string depend on the specific device and the various options available.

4.8.2 The Control Program

The original control program written for these experiments is called Mscan and is shown in a simplified form in Figure 4.13. The main function is far more complicated than that shown in the block diagram and is listed in Appendix A. The main function of Mscan has a menu that allows the user to set up a scan to measure the desired data. For the case of frequency dependent measurements, the scan would be set up to measure the impedance amplitude and phase as a function of frequency, sample temperature, and magnetic field. Mscan also establishes continuous monitoring of the sample temperature, liquid nitrogen level, and magnetic field level.

In order to control all of the devices, Mscan is broken down into three sublevels below the main routine. The first sublevel, getpar and setpar, permit access to reading and setting measurement values of all device operations with just the pair of functions. By routing most of the device communications through these two routines, a generic scan loop can be made in the main routine which can access any devices as set up through the main menu. This generic loop greatly simplifies the code used in the main routine. The second sublevel (shown in Figure 4.13 with MHP.C, Mtemp.C, and Mmagnet.C) contains all of the high level interface routines called by setpar and getpar to control each device. These routines create the ASCII strings that are to be sent to the devices and interpret the ASCII strings received from the devices. The third sublevel of routines is in MGEN.C and performs the low level access to the devices to transfer ASCII strings between the computer and measurement devices.

In order to have a reliable computer interface, it is crucial that the low level device interface functions work smoothly. The low level functions are listed in Appendix A.4. The most important of these functions are GEN_write() and GEN_read() which control the board functions to send or receive ASCII command strings. Although there are pre-written device functions that handle the operations performed by these "GEN" functions, it was found that the device functions become unreliable when trying to control several devices together. By writing the "GEN" functions to use only board commands, the program can fully control the addressing commands used in the interface line, thereby avoiding the problems found in the device functions.

Once the basic "GEN" routines are established, controlling the devices is just a matter of determining the control codes and parameters needed, assembling them into a character string, and calling the GEN_write() function. Some examples of these device control functions are shown in Appendix A.3. With the control functions, accessing each device becomes a simple matter of calling the function dedicated to the desired device command. For example, setting a current to 1mA and reading a voltage are accomplished by calling the functions: CURRENT_set(0.001); volts=NVMETER_read(). These simple commands are then assembled together into more complicated algorithms that are accessed within the getpar and setpar routines. The main routine of Mscan must control what parameters are to be set, record the data to be read, and maintain other basic tasks such as keeping the liquid nitrogen filled so that data can be taken accurately and efficiently.

Another program which is used in performing measurements at microwave frequencies is included in Appendix B. This program, named "Q_find," incorporates all of the low level device routines, but has an entirely different structure in the main routines. Because Q_find is used for only one type of measurement, namely measuring the Q of a

microwave cavity containing the sample to be measured,[19] all details of the measurement are coded into the main routine. There is no advantage in using the getpar and setpar routines in the Q_find program. Further description of Q_find and a partial program listing are included in Appendix B.

4.9 Conclusion

This chapter has discussed many of the basic experimental techniques and experimental systems used in the measurements on superconducting crystals. Sample preparation, sample probes, and the cryogenic dewars used have all been discussed, but the most difficult aspects of performing the actual measurements are to be discussed in Chapters 5-7. These coming chapters will describe how the electronic instrumentation is used to measure the impedance and susceptibility of the $YBa_2Cu_3O_7$ crystals and how the experimental considerations discussed in this chapter are important in performing these measurements.

References

[1] M. A. Beno, L. Soderholm, D. W. Capone, D. G. Hinks, J. D. Jorgensen, J. D. Grace, I. K. Schuller, C. U. Segre, and K. Zhang, Appl. Phys. Lett. **51**, 57 (1987).

[2] D. M. Newns, P. C. Pattnaik, and C. C. Tsuei, Phys. Rev. B **43,** 3075 (1991).

[3] R. G. Beck, D. E. Farrell, J. P. Rice, D. M. Ginsberg, and V. G. Kogan, Phys. Rev. Lett. **68**, 1594 (1992).

[4] S. Tajima, G. D. Gu, S. Miyamoto, A. Odagawa, and N. Koshizuka, Phys. Rev. B **48**, 16164 (1993).

[5] S. Jin, T. H. Tiefel, R. C. Sherwood, M. E. Davis, R. B. van Dover, G. W. Kammlott, R. A. Fastnacht, and H. D. Keith, Appl. Phys. Lett. **52**, 2074 (1988).

[6] M. Hawley, I. D. Raistrick, J. G. Beery, R. J. Houlton, Science **251**, 1587 (1991).

[7] A. Gupta, J. Z. Sun, and C. C. Tsuei, Science **265**, 1075 (1994); C. C. Tsuei, A. Gupta, and G. Trafas, Science **263**, 1259 (1994).

[8] B. D. Hunt, M. C. Foote, W. T. Pike, J. B. Barner, and R. P. Vasquez, Physica C **230,** 141 (1994); W. Chew, A. L. Riley, D. L. Rascoe, B. D. Hunt, M. C. Foote, T. W. Cooley, and L. J. Bajuk, IEEE Transactions on Microwave Theory And Techniques **39**, 1455 (1991).

[9] N.-C. Yeh, D. S. Reed, W. Jiang, U. Kriplani, F. Holtzberg, A. Gupta, B. D. Hunt, R. P. Vasquez, M. C. Foote, and L. Bajuk, Phys. Rev. B **45**, 5654 (1992).

[10] D. L. Kaiser, F. Holtzberg, M. F. Chisholm, and T. K. Worthington, Journ. Cryst. Growth **85,** 593 (1987).

[11] M. Z. Cieplak, G. Xiao, C. L. Chien, A. Bakhshai, D. Artymowicz, W. Bryden, J. K. Stalick, and J. J. Rhyne, Phys. Rev. B **42**, 6200 (1990).

[12] R. J. Cava, B. Batlogg, C. H. Chen, E. A. Rietman, S. M. Zahurak, and D. Werder, Nature **329**, 423 (1987).

[13] B. G. Hyde, J. G. Thompson, R. L. Withers, J. G. Fitzgerald, A. M. Stewart, D. J. M. Bevan, J. S. Anderson, J. Bitmead, and M. S. Peterson, Nature **327**, 402 (1987).

[14] A. H. Morrish, X. Z. Zhou, M. Raudsepp, I. Maartense, J. A. Eaton, and Y. L. Luo, Canadian Journ. of Phys. **65**, 808 (1987).

[15] R. P. Vasquez, M C. Foote, L. Bajuk, and B. D. Hunt, J. Electron Spectrosc. Relat. Phenom. **57,** 317 (1991).

[16] *Lake Shore Product Catalog and Reference Guide*, (Lake Shore Cryotronics, Inc., Westerville, 1992).

[17] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. Lett. (in press).

[18] *IEEE Standard Digital Interface for Programmable Instrumentation*, IEEE-488 st. 488 (1978); *NI-488 MS-Dos Software Reference Manual*, Part #320135-01 (National Instruments Corp., Austin, 1988), Appendix E.

[19] N.-C. Yeh, U. Kriplani, W. Jiang, D. S. Reed, D. M. Strayer, J. B. Barner, B. D. Hunt, M. C. Foote, R. P. Vasquez, and A. Kussmaul, Phys. Rev. B **48**, 9861 (1993); U. Kriplani, N.-C. Yeh, W. Jiang, D. S. Reed, A. Gupta, and A. Kussmaul, *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p.795.

Chapter 5 : AC Impedance Measurements

Direct transport measurements of the ac impedance of a sample involve passing a uniform current $I = I_0 e^{i2\pi ft}$ with a fixed amplitude I_0 and a fixed frequency f through the sample and measuring the resulting voltage $V = V_0 e^{i2\pi ft}$ across it. The impedance, defined by $Z \equiv \rho e^{i\Phi} \equiv V/I$, will be current independent provided the current is sufficiently small to remain in the linear response regime of the sample. Generally speaking, frequency dependent transport measurements are better than dc current-voltage characteristics in providing more conclusive evidence for a second-order phase transition, because as pointed out in references 1 and 2, scaling of the dc measurements over a limited current range is not distinguishable from the predicted behavior of conventional flux creep theory. In addition, this direct transport measurement has advantages over the more commonly performed magnetic susceptibility measurements because it is a bulk measurement with a uniform current through the sample, rather than a surface current limited by the magnetic penetration depth λ. The surface current is much more difficult to interpret quantitatively and is more sensitive to surface defects. However, because of the experimental difficulties involved in studying the frequency dependence of very small signals over a broad frequency range, only one previous experiment before our work has been performed to measure the frequency dependent impedance of high temperature superconductors. That previous work and the problems in its experimental design are discussed in section 5.1. Section 5.2 describes the experimental setup used for this work and how it overcomes the previous limitations. The data in section 5.3 demonstrate the success of this technique and

shows critical exponents and transition temperatures which are consistent with previous dc measurements, giving strong evidence for the existence of the second-order phase transition and the validity of the critical exponents found in this work. In addition, the development of this technique also provides a useful tool for future investigations of vortex phase transitions in any type-II superconductors.

5.1 Previous Experimental Work

The difficulties involved in performing low level impedance measurements over a broad frequency range have limited the number of previous attempts to perform these experiments[3]. Only one experiment has been published before our work which measures the ac impedance of a superconductor over a broad frequency range (f=10kHz $\rightarrow$ 500MHz), and that work suffers from several experimental complications.[4] In that experiment, impedance measurements are made on a patterned thin film of $YBa_2Cu_3O_7$ using a network analyzer. The network analyzer, as shown in Figure 5.1, transmits a sinusoidal voltage through the coaxial cable which is terminated by the sample. The reflected and transmitted components of the signal are measured and used to calculate the impedance for the sample. This technique works only because the impedance mismatch between the cable and sample causes the transmitted component to vary with frequency and sample impedance from full reflection to full transmission. The variation in transmitted power leads to a variation in the current from nearly 0 to some maximum current I_{max}. Reference 4 claims that the maximum applied current density of $I_{max}=5\times10^6 Am^{-2}$ is sufficiently small and the frequency sufficiently large that the measurements are taken in the linear response regime for $T \geq T_M$ and the impedance is independent of the applied current. However, for

$T<T_M$ or any other investigation into the nonlinear properties of superconductors, the variations in current amplitude would severely distort the experimental results.

Although network analyzers can measure impedances over a very broad frequency range, they are limited in sensitivity when the sample impedance is far from the characteristic impedance of the cable[6] (typically ~50Ω). This limitation is evident in reference 4 which shows the amplitude resolution limited to 0.1Ω and the phase measurements limited to frequencies above 10^7Hz. The limited amplitude resolution is compensated in reference 4 by using a film patterned into a geometry which greatly enhances the resistance to 700Ω in the normal state (as compared to 0.03Ω for a typical single crystal). However, increasing the resistance has the disadvantage of increasing the effects of the stray capacitance "short" in parallel with the sample. This stray capacitance is always present due to a capacitive coupling between the current input at the sample and ground. For example, two contact wires on a sample with a diameter of 0.4mm that extend for a length of 1 cm at a distance of 3mm apart would lead to a stray capacitance of $C_{stray}=2\times10^{-13}$F. The impedance of such a capacitive coupling is given by $Z_{stray} = 1/\left(2\pi f C_{stray}\right)$, where f is the frequency of the applied signal. At the applied frequency of f=500MHz, the impedance is $Z_{stray}\approx1600\Omega$, which is comparable to the film resistance, thereby providing an alternate current path which severely distorts the measured signal.

Another serious limitation to the data presented in reference 4 is that measurements are made only in a single magnetic field of 5kOe, preventing any consistent verification of the universality of the critical exponents and vortex transport functions. Furthermore, the "vortex-glass" transition temperature was not obtained directly from these ac

measurements, but was obtained from separate dc measurements without proper critical scaling analysis so that large variations in the "critical exponents" were found when the magnetic field changes from 5kOe to 40kOe[5]. Without a self consistent procedure to identify the transition temperature $T_M(H)$, the corresponding estimate of the critical exponents in reference 4 becomes very questionable. With the limited data, inconsistent analysis, and potentially serious problems with the data near T_M due to unknown fluctuations in the applied current, it is difficult to draw conclusions from reference 4 of the validity of the scaling analysis or the accuracy of the exponents obtained.

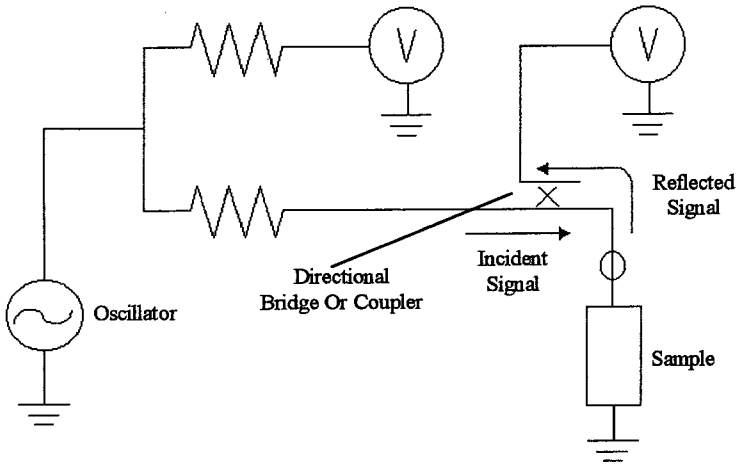


Figure 5.1. Block diagram of the network analysis method[6]. The sample impedance is calculated from the reflection coefficient, which is obtained by measuring the ratio of the incident signal to the reflected signal. A directional coupler or bridge is used to detect the reflected signal.

5.2 Experimental Setup

In order to overcome the problems inherent in the network analyzer, we use the HP4194A impedance analyzer to perform the frequency dependent measurements in the configuration shown in Figure 5.2. Although the maximum usable frequency is slightly lower (f_{max}~10MHz), the minimum usable frequency is considerably lower (f_{min}~100Hz) so that the number of decades in frequency measurements is actually increased. To perform measurements, the impedance analyzer applies an ac voltage $\left(V_{ac} = V_0 e^{i2\pi ft}\right)$ with constant amplitude $V_0 = 0.5\,\mathrm{V}$ on the high output, which by using a non-inductive load resistor

$R_L = 1\text{k}\Omega$ in series with the sample provides an ac current with an amplitude of $500\mu\text{A}$. The amplitude of this current will remain constant as long as the sample impedance Z_{samp} (including contact resistance) is much less than the load resistor $\left(|Z_{samp}| << R_L\right)$ so that $I_{ac} = V_{ac}/\left(Z_{samp} + R_L\right) \approx V_{ac}/R_L$. The amplitude of the current I_{ac} is continually monitored by the HP4194A and is found to be constant for frequencies f<~300kHz with a

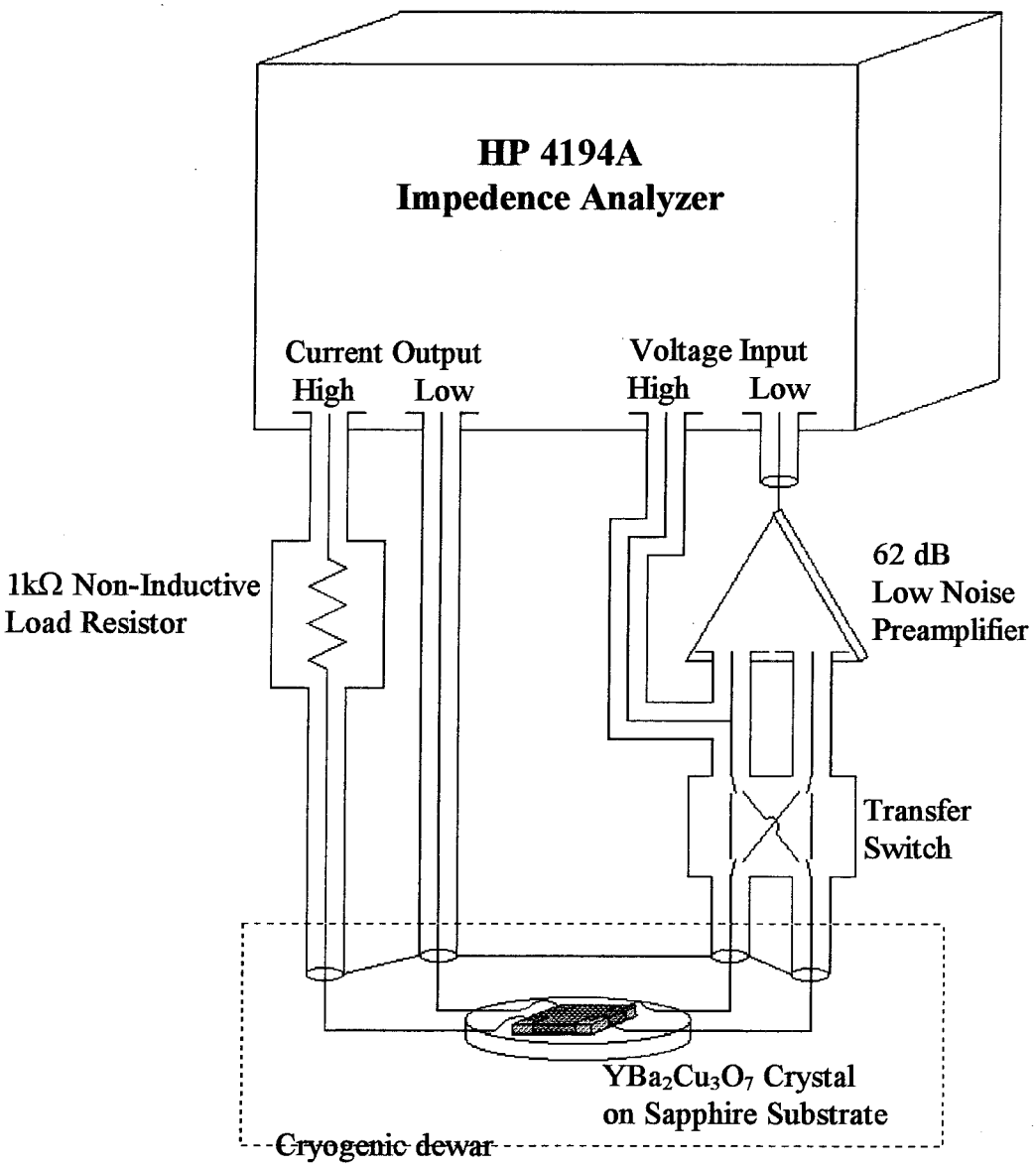


Figure 5.2. Block diagram of the ac impedance measurement system.

change of $\Delta I_{ac}<10\%\times I_{ac}$ at the highest frequency of $f\approx 5$MHz. For the $YBa_2Cu_3O_7$ crystal used in these measurements, the current $I_{ac}\approx 500\mu$A corresponds to a current density of only $10^4 Am^{-2}$, which is found to be a sufficiently low current density to remain in the linear response regime of the sample for most temperatures of interest. Because of the small current used and the small resistance of a $YBa_2Cu_3O_7$ single crystal, the HP4194A's sensitivity of 0.1mV is clearly inadequate. However, by incorporating a low noise preamplifier in front of the voltage input on the impedance analyzer, the minimum measurable signal level is 10nV which corresponds to a resistance of $2\times 10^{-5}\Omega$ and is sufficiently sensitive to provide useful information.

Before discussing the preamplifier and how it is connected to the HP impedance analyzer, it is necessary to explain exactly how the impedance analyzer performs measurements. The HP impedance analyzer uses a relatively standard auto-balancing bridge and lock-in technique as shown in Figure 5.3 to achieve accurate frequency dependent

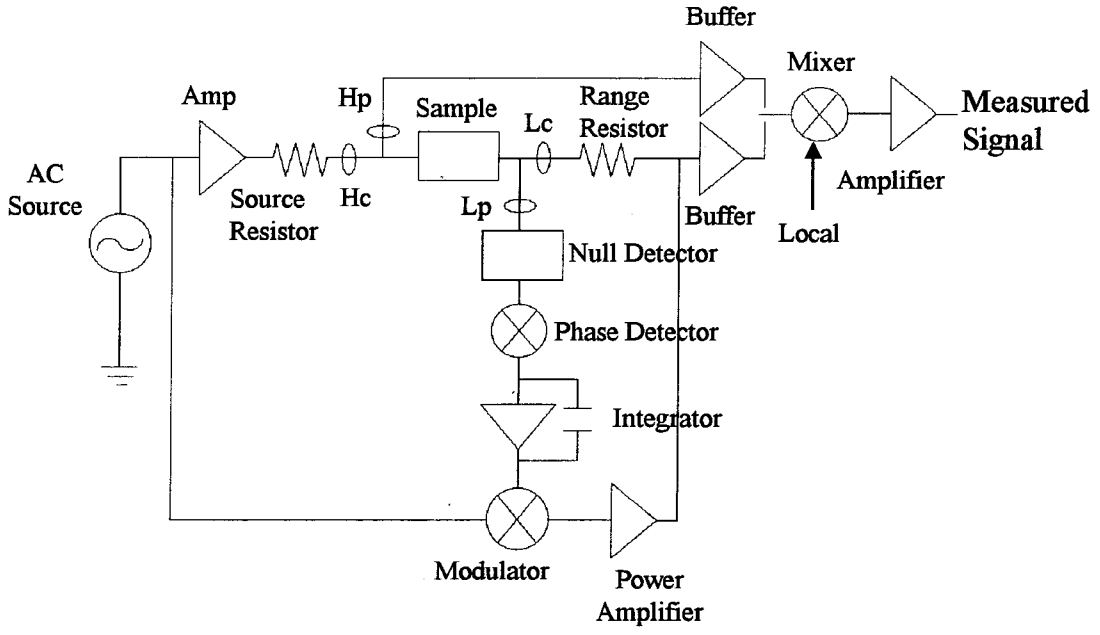


Figure 5.3. Schematic of the Measurement section of the HP 4194A Impedance analyzer.

measurements. One of the key points in the circuit is that the low current output is not a ground, but is a current output that is actively phase adjusted and amplitude adjusted to make the low voltage exactly zero. The null detector connected to the low voltage input adjusts the phase and amplitude of the current being output through the low current output until it is equal to (but 180° out of phase with) the high output current, resulting in zero voltage at the null detector. By using this auto-balancing bridge method rather than a conventional differential input found in most amplifiers, the common mode signal of the voltage reading is eliminated and the contact resistance from the low voltage connection is eliminated because no current flows through the low voltage cable. For measurements of large signals, the only relevant voltage is the high voltage input. However, the null detector only detects voltages >~0.01mV, so to measure sample voltages of ~10nV it is still necessary to amplify the difference between the high sample voltage and the "null" voltage as shown in Figure 5.2. The additional connection of the low sample voltage to the low voltage input of the impedance analyzer is necessary to enable the null detector to properly adjust the low current output. The output voltage of the amplifier is read at the high voltage input on the impedance analyzer and is mixed with the reference signal in the manner of a standard lock-in amplifier. The in phase and out of phase components of the signal can be recorded by the impedance analyzer and are subsequently transferred from the HP4194A to a computer.

One of the few commercial amplifiers available which cover a frequency range up to 10MHz with the needed 10nV noise level is an amplifier built by Analog modules, Inc. Their model 322-10 amplifier has a bandwidth of 100Hz to 45MHz, a noise level of only

$280\,\mathrm{pV}/\sqrt{\mathrm{Hz}}$, and a gain of 40dB. However, the input impedance of these amplifiers is only 50Ω to 150Ω, which is comparable to the contact resistance at low temperatures for some samples (~20Ω). Such a low input impedance on the amplifier nullifies the advantage of using the 4-point measurement technique and introduces the contact resistance as a significant distortion in the signal. To see the effects of contact resistance more clearly, consider the diagram in Figure 5.4. The desired sample signal is $V_+ - V_-$, but the measured signal is $V_{out} = V_{M+} - V_{M-}$, where $V_{M\pm} = V_\pm \left[Z_{amp} \Big/ \left(Z_{C\pm} + Z_{amp} \right) \right]$ and $Z_{C\pm}$ is the contact impedance on the sample and Z_{amp} is the amplifier input impedance. Therefore, the measured signal becomes

$$V_{out} = \frac{V_+}{\left(\frac{Z_{C+}}{Z_{amp}} \right) + 1} - \frac{V_-}{\left(\frac{Z_{C-}}{Z_{amp}} \right) + 1}. \qquad (5.1)$$

For $Z_{amp} >> Z_{C\pm}$, Eq. (5.1) clearly reduces to $V_{out} = V_+ - V_-$, but if $Z_{amp} \approx Z_{C\pm}$ the results could be severely distorted. Distortion to the measured

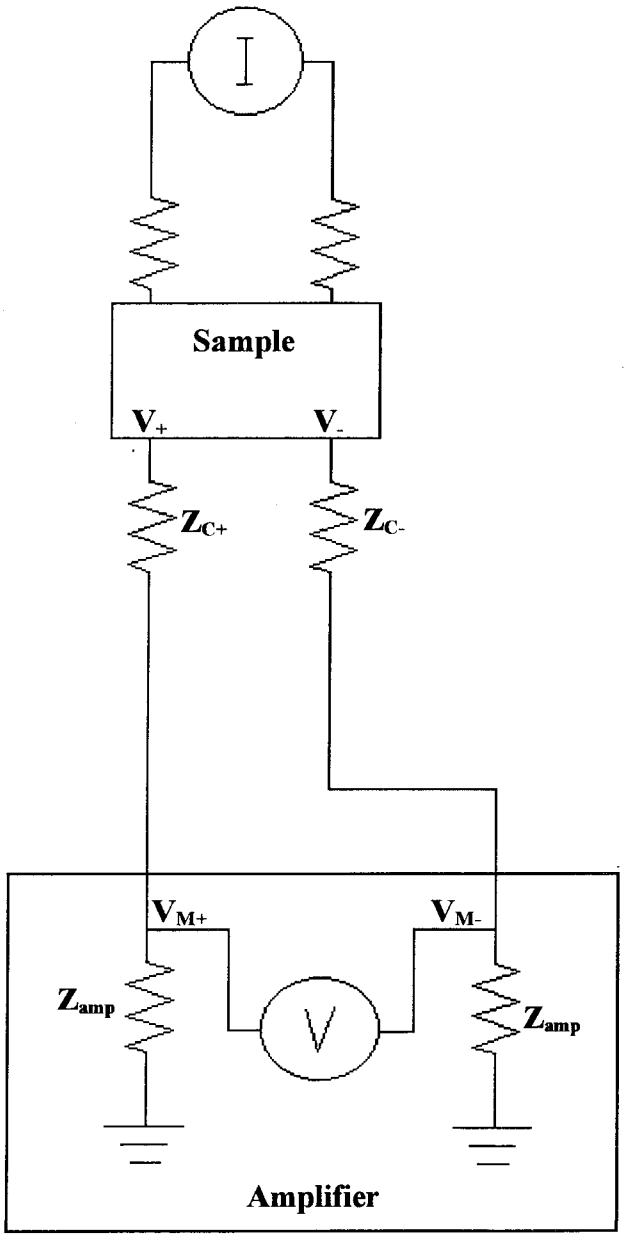


Figure 5.4. The problem of contact resistance in measurements.

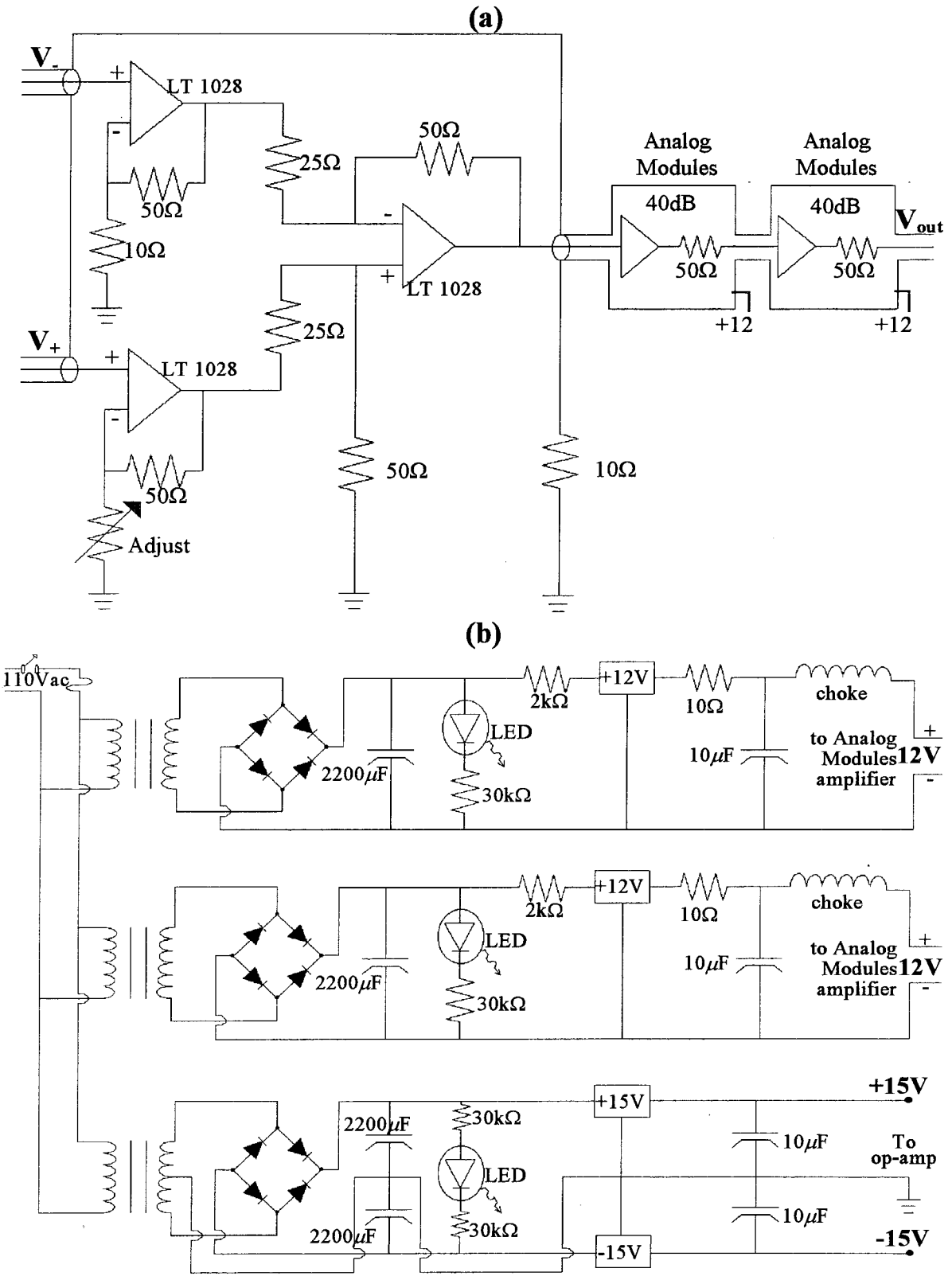


Figure 5.5. Preamplifier used in ac impedance and susceptibility measurements. **(a)** The differential amplifier is built with an initial stage of LT 1028 op-amps which feed the signal into two additional amplifiers built by Analog Modules, Inc. **(b)** The power supply is designed to minimize cross talk between amplifier segments, and filtering is added to minimize the external noise entering the system.

signal becomes very significant if $Z_{C+} \neq Z_{C-}$ and $V_{\pm} >> V_{+} - V_{-}$, which is nearly always true, especially when the sample is in the superconducting state. Furthermore, the contact resistance is strongly dependent on material properties of the sample and could show very different frequency dependent properties for the sample in the normal state versus the sample in the superconducting state. It is therefore essential that $Z_{amp} >> Z_{C\pm}$.

In order to establish a high input impedance on the preamplifier, the high impedance input stage shown in Figure 5.5 was constructed to precede the Analog Modules amplifiers. With an initial input stage, the full benefit of the low noise level of the Analog Modules amplifier cannot be used, but the large bandwidth and high gain are still needed for the secondary stages. The LT 1028 op-amps used in the circuit have an input impedance of 20kΩ, a noise level of only $0.85\text{nV}/\sqrt{\text{Hz}}$ at 10kHz and a frequency response which is linear up to ~2MHz. With this preamplifier, signals as low as 10nV can be measured, which corresponds to a resistance of $20\mu\Omega$ at the applied current of 500μA. Measurements at frequencies below ~2MHz show little distortion from the amplifier, though for frequencies in the range 2MHz → 10MHz, the amplifier begins to show significant phase lag and the amplitude of the gain begins to drop dramatically. Beyond 10MHz, the amplifier cannot be used. Although the amplifier is currently the primary limitation to higher frequency measurements, other factors make the system unable to perform beyond ~10MHz. One problem that occurs at high frequencies is that the rf wavelength becomes comparable to the cable lengths used so the cables must be treated as transmission lines rather than idealized conductors. A common estimate for the frequency at which transmission line effects become important is when the cable length corresponds to 1/8 wave-

length. For a cable length of 3m and a propagation speed of 0.71c (where c is the speed of light in vacuum), the 1/8 wavelength corresponds to a frequency of 8.9MHz. Therefore, at frequencies $f >\sim 9\,\mathrm{MHz}$, impedance mismatches will cause signals to be partially reflected so the applied current will not be constant and the voltage readings will be distorted. Additionally, the relevant input impedance Z_{amp} used in Eq. (5.1) will no longer be the amplifier input impedance, but the 50Ω characteristic impedance of the transmission line, so that the contact resistance will begin to affect the signal. By limiting the frequency range to $f <\sim 2\,\mathrm{MHz}$, these problems are not completely eliminated, but are reduced and can be compensated with an accurate calibration. The calibration also compensates for the stray inductance and capacitance which lead to an additional background signal for frequencies $f >\sim 50\,\mathrm{kHz}$.

Parasitic inductance and capacitance are sources of error in the measurements which are unavoidable in any circuit components and should give a purely imaginary background signal Z_{back}. Additionally, the preamplifier has a frequency dependent gain $G(f)$, and along with the cables introduces a phase delay $\theta(f)$. Therefore, the measured signal is of the form $Z_{meas} = \left(Z_{signal} + Z_{back}(f)\right)G(f)e^{i\theta(f)}$, where Z_{signal} is the desired signal from the sample. These background parameters are nearly independent of magnetic field and temperature, but show a strong frequency dependence so that they must be evaluated at every frequency to be measured. To find $G(f)$, the normal state sample signal (Z_{signal} for $T >> T_M$) is assumed to be independent of frequency for $f < 2\,\mathrm{MHz}$, and is also assumed to be much larger than the background $\left|Z_{signal}\right| >> \left|Z_{back}\right|$. Thus $G(f)$ is

found from $|Z_{meas}(f)| = |Z_{signal}(f=0)|G(f)$ at $T>>T_M$, where $|Z_{signal}(f=0)|$ can be found accurately from dc measurements. The phase adjustment $\theta(f)$ can be found in a similar manner by assuming that the sample signal for $T << T_M$ is purely imaginary so that $(Z_{signal} + Z_{back}) = |Z_{signal} + Z_{back}|e^{i\pi/2}$, giving $Z_{meas}(f) = |Z_{signal} + Z_{back}|G(f)e^{i(\theta(f)+\pi/2)}$ for $T<<T_M$. The phase shift $\theta(f)$ is then just the measured phase shifted by $\pi/2$. Finally, Z_{back} can be found by measuring a short such that $Z_{signal} \approx 0$. An effective short can be made with a small piece of gold foil of approximately the same size as the sample with contacts and wiring arranged in the same manner as that used for the crystal. The gold foil is considerably thicker than the crystal and can be made to have a resistance $\approx 20\mu\Omega$, corresponding to a voltage ≈10nV, which is the instrumental sensitivity. With the gold short in place, the measured signal is given by $Z_{meas}(f) = Z_{back}(f)G(f)e^{i\theta(f)}$, so all parameters can be found.

One peculiar feature which appeared in the data that could not be corrected by the calibration described above is an occasional step in the impedance versus frequency data so that the measured signal is given by $Z_{meas} = Z_{samp} + Z_{step}$ where Z_{samp} is the sample impedance and Z_{step} is the artificial step. This step is apparently due to an instability in the HP4194A such as that found when switching range resistors in the output current. By reversing the voltage leads with the transfer switch shown in Figure 5.2, these steps are found to be unaffected while the signal voltage shifts in phase by 180°, so the measured signal becomes $Z'_{meas} = -Z_{samp} + Z_{step}$. Because the step is unaffected by reversing the voltage leads, it must be an artifact generated within the impedance analyzer itself and not

the result of the sample signal or external circuits. The step is accurately reproduced in measurements with the transfer switch forward (Z_{meas}) and reversed (Z'_{meas}), so the actual signal is easily obtained by subtracting these two measurements, $(Z_{meas} - Z'_{meas})/2 = Z_{samp}$.

The measurements are performed on a $YBa_2Cu_3O_7$ single crystal with dimensions of 2mm×3mm×20μm with the shortest dimension along the c-axis. Details of the sample preparation and procedure for making electrical contacts are discussed in Chapter 4. The sample is cooled with the magnetic field turned off from a temperature well above the superconducting phase transition to the lowest temperature to be measured. With the temperature stabilized at the lowest temperature to be measured, the dc magnetic field is applied with $\vec{H} \parallel$ crystal c-axis. Using the frequency sweep capability of the HP4194A impedance analyzer, impedance measurements are made at 400 different frequencies from 100Hz to 2MHz. The transfer switch then reverses the voltage leads and the frequency sweep is repeated. These two sets require approximately 20 minutes, so the temperature must remain stable to within ±0.003K for that length of time. The temperature setpoint is then increased by ≈0.02K for the next temperature to be measured. After waiting for the temperature to stabilize, the set of frequency sweeps is repeated. The frequency sweep is repeated for a series of temperatures crossing the vortex-glass melting temperature to give a series of isotherms as shown in Figure 5.6 and Figure 5.7. The entire process is repeated for different magnetic fields from 1kOe to 90kOe.

5.3 Experimental Results

The data presented here are composed of measurements made in nine different magnetic fields as indicated in Table 5-1. Figure 5.6 shows the resistivity amplitude versus frequency along isothermal curves for various dc magnetic fields, and the corresponding phase is shown in Figure 5.7. The scaling functions as derived in Chapter 3 for the amplitude ρ and phase Φ are given in terms of the scaled frequency $\tilde{f}$ by

$$\begin{aligned} \tilde{\rho}_{\pm}(\tilde{f}) &\equiv \rho\left|1-(\mathrm{T}/\mathrm{T_M})\right|^{-\nu(z-1)} \\ \tilde{\Phi}_{\pm}(\tilde{f}) &\equiv \Phi \\ \tilde{f} &\equiv f\left|1-(\mathrm{T}/\mathrm{T_M})\right|^{-\nu z}, \end{aligned} \tag{5.2}$$

where ν and z are the static and dynamic exponents respectively, and $\tilde{\rho}_{\pm}$ and $\tilde{\Phi}_{\pm}$ are the scaling functions for $T<T_M$ (-) and $T>T_M$ (+).

The insets of Figure 5.6 and Figure 5.7 show the universal functions $\tilde{\rho}_{\pm}(\tilde{f})$, and $\tilde{\Phi}_{\pm}(\tilde{f})$ obtained by using Eq. (5.2) and selecting values of ν, z, and T_M that best collapse all of the isothermal curves with temperatures within the critical regime into a single curve.

	Amplitude			Phase		$\tilde{\rho}_{-}(\tilde{f})$	
H(kOe)	T_M(K)	ν	z	$\Phi(T_M)$	z	slope (x)	z
1	92.51±0.04	0.70±0.10	3.4±0.2	50°±10°	2.4±0.6	0.70±0.05	3.3±0.5
3	91.77±0.01	0.80±0.20	3.0±0.3	55°±8°	2.6±0.6	0.68±0.04	3.1±0.4
5	91.21±0.01	0.70±0.03	3.0±0.2	55°±10°	2.6±0.7	0.62±0.05	2.6±0.5
6	90.93±0.01	0.60±0.10	3.0±0.2	59°±7°	2.9±0.7	0.67±0.05	3.0±0.5
10	90.04±0.01	0.60±0.10	3.0±0.5	59°±8°	2.9±0.7	0.67±0.03	3.0±0.5
30	85.98±0.01	0.70±0.10	2.8±0.5	60°±8°	3.0±0.7	0.66±0.02	2.9±0.5
50	82.62±0.01	0.70±0.05	3.0±0.2	58°±5°	2.8±0.4	0.66±0.03	2.9±0.3
70	79.34±0.05	0.65±0.20	2.8±0.8	60°±7°	3.0±0.7	0.61±0.04	2.6±0.3
90	75.85±0.03	0.80±0.20	3.0±0.5	59°±9°	2.9±0.8	0.65±0.01	2.9±0.1

Table 5-1. The critical exponents ν and z obtained from "collapsing" the amplitude (ρ), the phase (Φ) measured at $T_M(H)$, and the slope of the universal function $\tilde{\rho}_{-}(\tilde{f})$, for fields from 1 to 90 kOe.

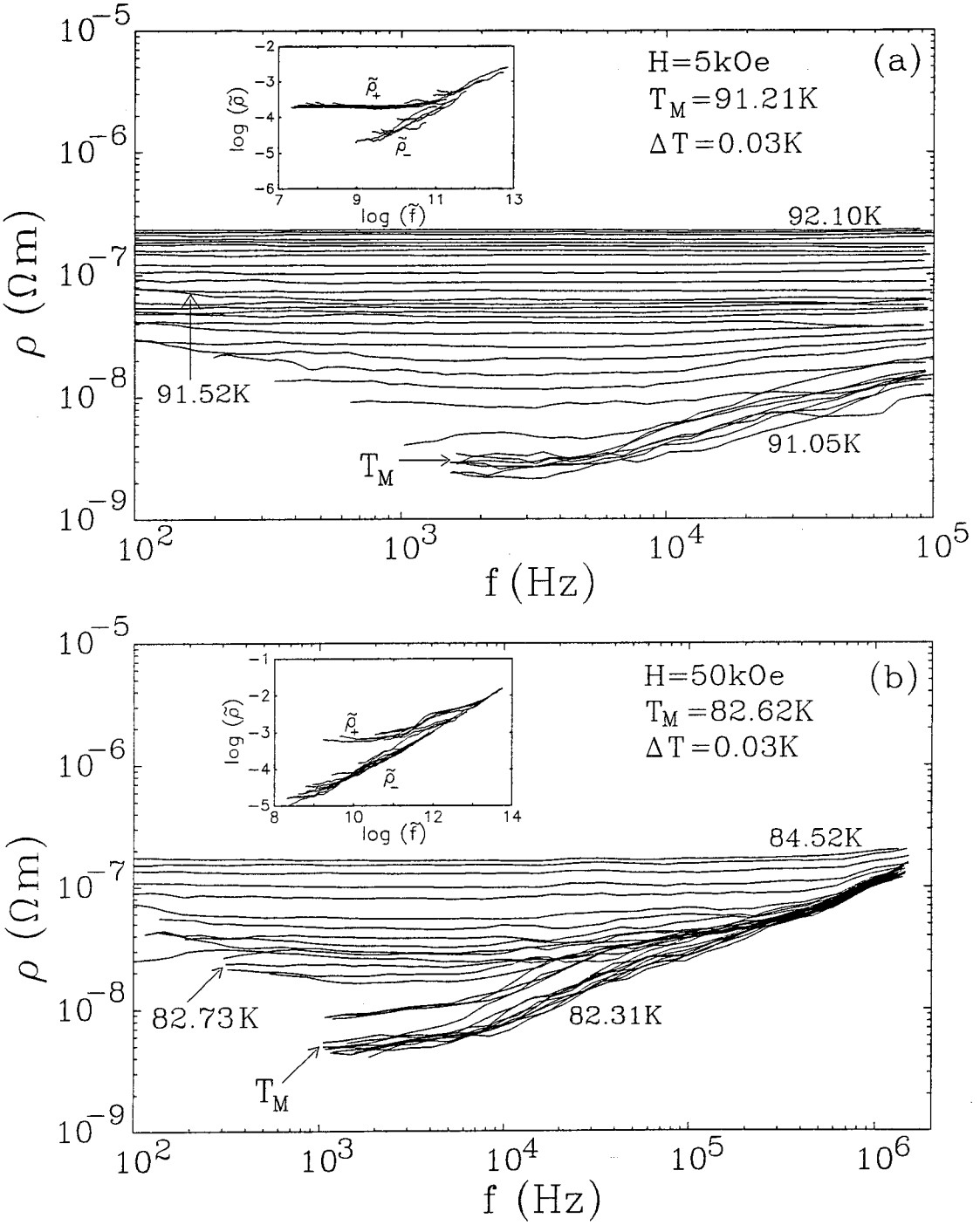


Figure 5.6. Impedance (ρ) vs. frequency (f) isotherms for H||c-axis and **(a)** H=5kOe, **(b)** H=50kOe , and **(c)** (next page) H=10 kOe. The insets are the universal functions $\tilde{\rho}$ vs. $\tilde{f}$ obtained from "collapsing" the isotherms using Eq. (5.2) with values of ν, z, and T_M as shown in Table 5-1. The temperature increment ΔT between successive traces is 0.03K, and the critical regime is indicated by the temperature interval between the lowest temperature shown above and the arrow.

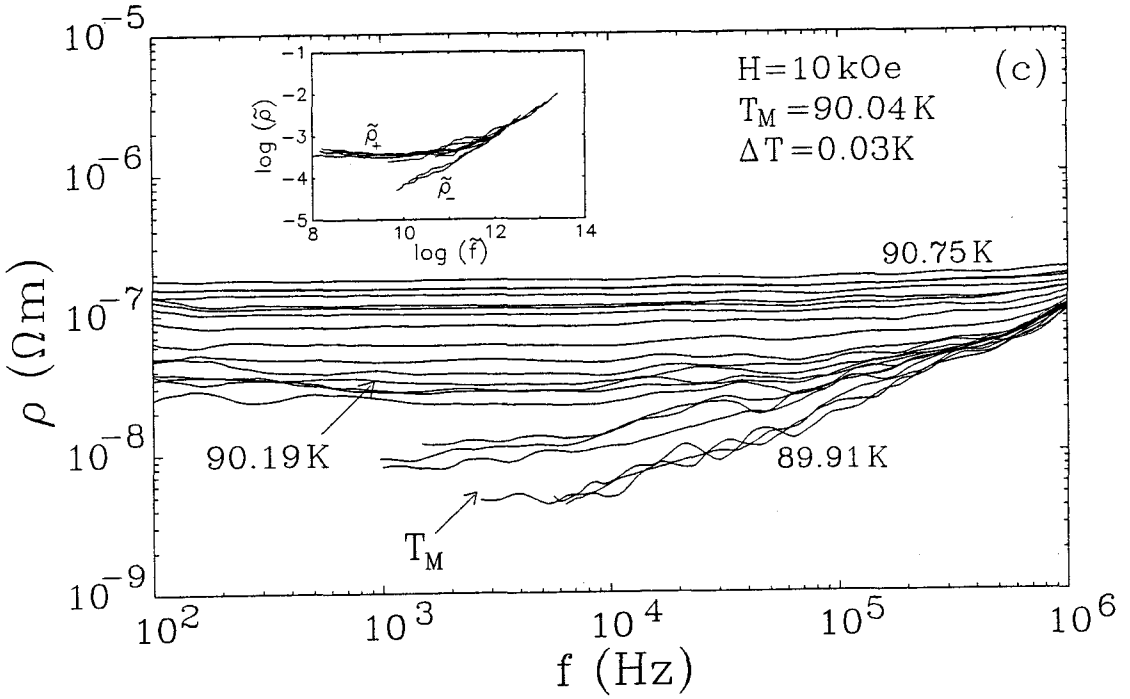


Figure 5.6(c). Impedance (ρ) vs. frequency (f) isotherms for H||c-axis and (c) H=10 kOe. The insets are the universal functions $\tilde{\rho}$ vs. $\tilde{f}$ obtained from "collapsing" the isotherms using Eq. (5.2) with values of ν, z, and T_M as shown in Table 5-1. The temperature increment ΔT between successive traces is 0.03K, and the critical regime is indicated by the temperature interval between the lowest temperature shown above and the arrow.

The values obtained by "collapsing" the resistivity isotherms are shown under the "Amplitude" section of Table 5-1. In the asymptotic limit of $\tilde{f} \to \infty$, $\tilde{\rho}(\tilde{f})$ becomes a power law $\tilde{\rho} \propto \tilde{f}^{\,x}$, where $x = 1 - 1/z$. A measure of the slope of $\log(\tilde{\rho})$ vs. $\log(\tilde{f})$ and the corresponding z are listed in Table 5-1. The consistency of the slope in all magnetic fields demonstrates the universality of the scaling function $\tilde{\rho}(\tilde{f})$ and provides a self consistent verification of the value obtained for z. In the limit of $T \to T_M$, $\rho(f)$ will follow the same power law as $\tilde{\rho}(\tilde{f})$, so that $\rho(f, T \to T_M) \propto f^{\,x}$.

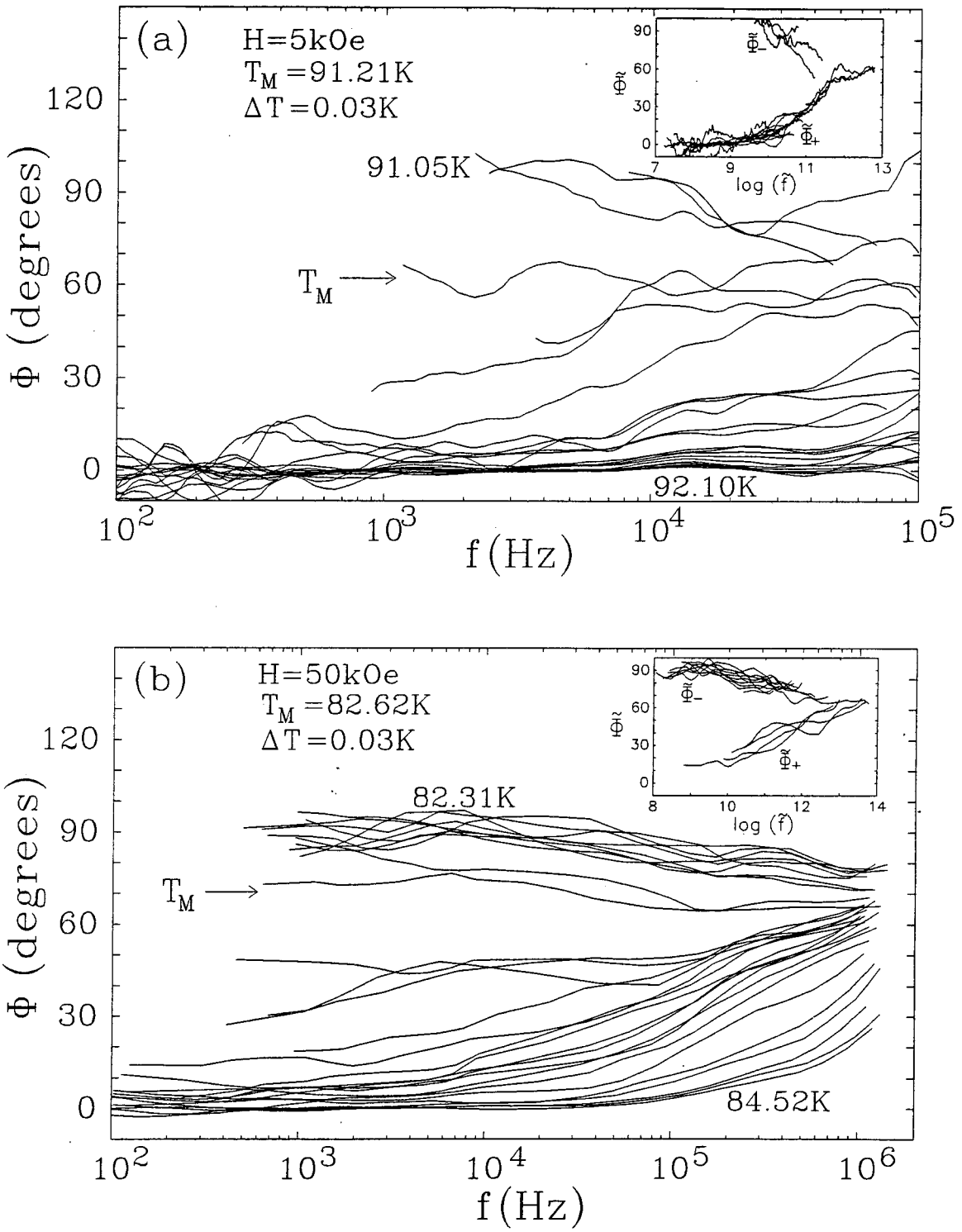


Figure 5.7. Phase (Φ) vs. frequency (*f*) isotherms for *H*||c-axis and **(a)** *H*=5kOe, **(b)** *H*=50kOe, and **(c)** (next page) H=10kOe. The insets are the universal functions $\widetilde{\Phi}$ vs. $\widetilde{f}$ obtained from "collapsing" the isotherms. The temperature increment ΔT is 0.03K.

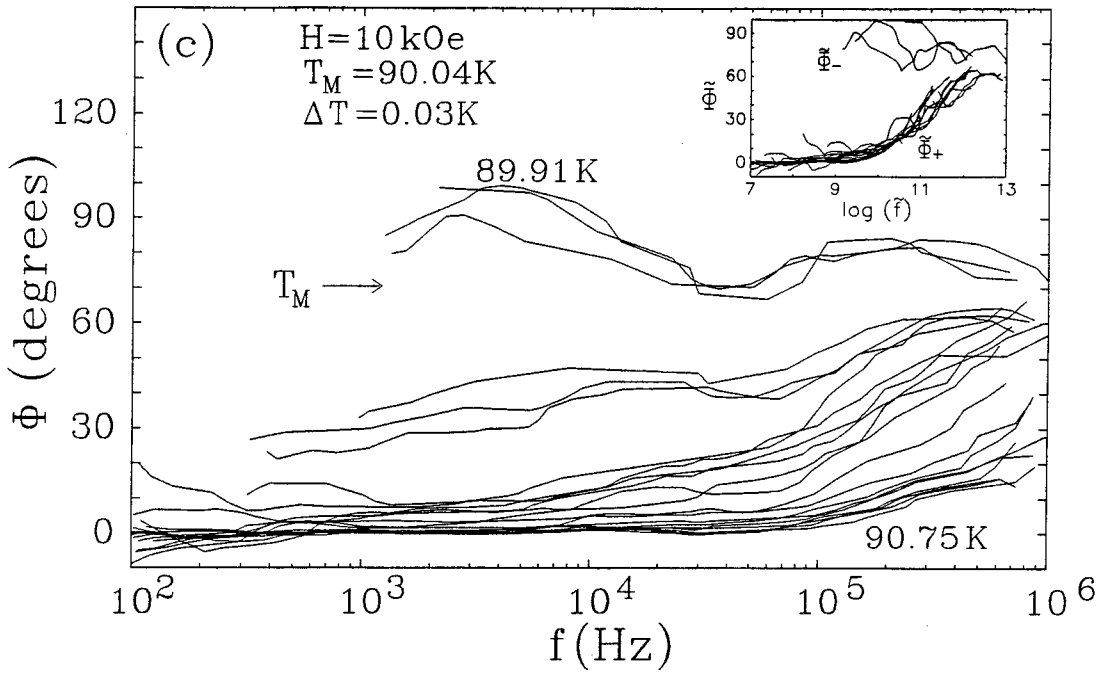


Figure 5.7(c). Phase (Φ) vs. frequency (f) isotherms for H||c-axis and (c) H=10 kOe. The insets are the universal functions $\tilde{\Phi}$ vs. $\tilde{f}$ obtained from "collapsing" the isotherms. The temperature increment ΔT is 0.03K.

By inserting the phase and amplitude of the resistivity into the Kramers-Kronig relation[7],

$$\Phi = -\mathcal{P}\int_{-\infty}^{\infty}\frac{df'}{\pi}\frac{\ln|\rho(f')|}{f'-f} \tag{5.3}$$

where $\mathcal{P}$ denotes the principle value of the integration, we obtain a frequency independent phase at T_M: $\Phi(T_M) = \frac{\pi}{2}\left[1-\frac{1}{z}\right]$. Although this frequency independent phase can be seen in Figure 5.7, it is more easily seen in Figure 5.8 where the 50kOe data and 10kOe data are converted to phase vs. temperature for different frequencies from 1kHz to 2MHz. Figure 5.8 clearly shows the phase at all frequencies merging to the same value at the melting temperature. The phase $\Phi(T_M)$ evaluated in as in Figure 5.8 for all fields is listed

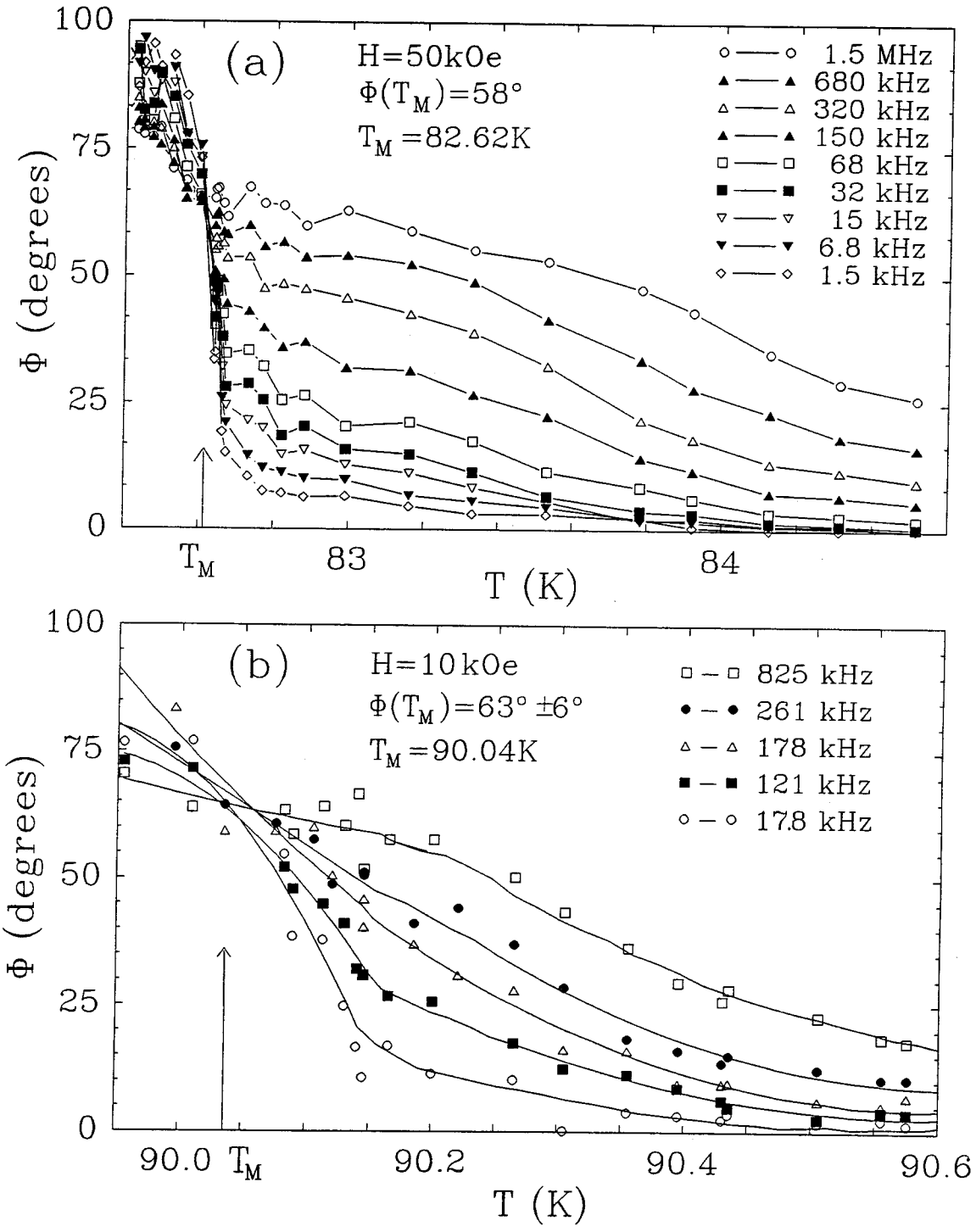


Figure 5.8. Phase (Φ) vs. temperature (T) data converted **(a)** from Figure 5.7(b) with H=50kOe and **(b)** from Figure 5.7(c) with H=10kOe for several frequencies by averaging points of each isotherm in a small interval about the indicated frequency.

in Table 5-1, as well as the value of z obtained from Eq. (5.3), is consistent with the values of z found by the other methods within experimental accuracy. The exponents obtained here are also consistent with results of dc current-voltage measurements which give[8] ν=0.67±0.1 and z=3.0±0.2.

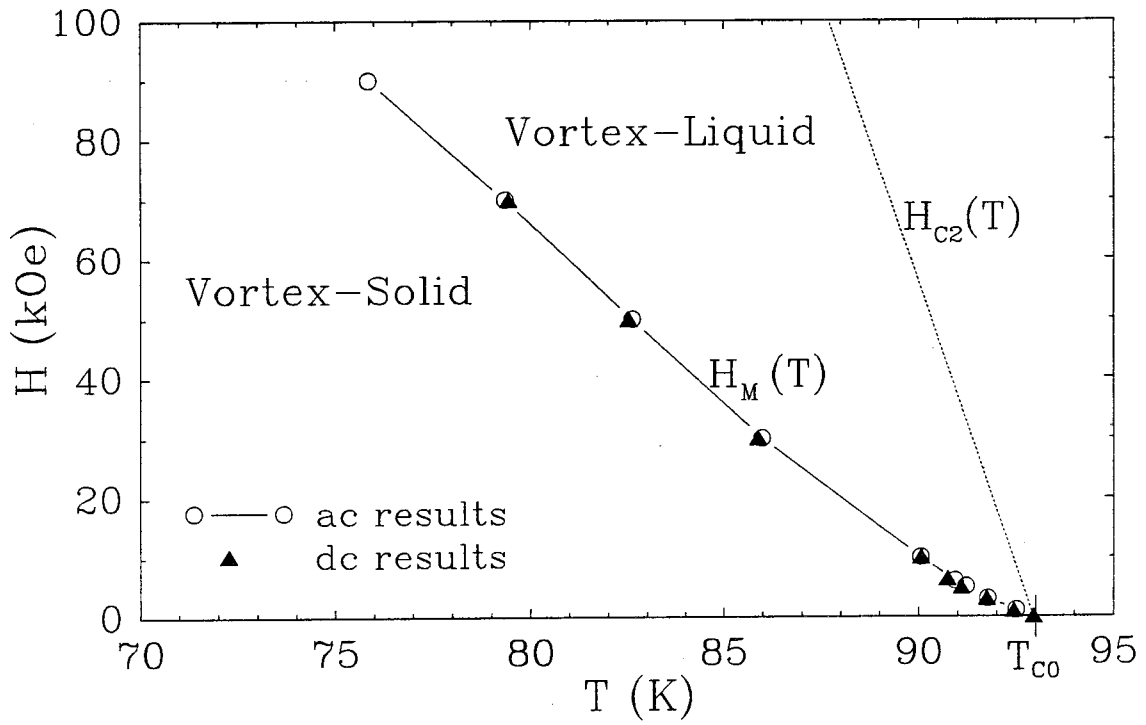


Figure 5.9. The vortex phase diagram for H||c-axis. The line H_M(T) is derived from the scaling fits in Figure 5.6. The dc data (reference 8) are shown to emphasize the consistency between two different measurement techniques. The dashed line $H_{c2}(T)$ is the upper critical field.

With the ac transport measurements, universal scaling analysis has been consistently demonstrated over two decades in magnetic field and two independent parameters, the amplitude and phase. These experiments have shown consistent scaling functions and nearly identical values for the critical exponents. Combining the ac measurements with the same results from dc current-voltage characteristics and the absence of any alternative theoretical explanation, it lends conclusive support for a second-order vortex-solid melting transition.

References

[1] E. Zeldov, N. M. Amer, G. Koren, A. Gupta, R. J. Gambino, and M. W. McElfresh, Phys. Rev. Lett. **62,** 3093 (1989).

[2] S. N. Coppersmith, M. Inui, and P. B. Littlewood, Phys. Rev. Lett. **64,** 2585 (1990).

[3] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6150 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 413.

[4] H. K. Olsson, R. H. Koch, W. Eidelloth, and R. P. Robertazzi, Phys. Rev. Lett. **66**, 2661 (1991).

[5] R. H. Koch, V. Foglietti, W. J. Gallagher, G. Koren, A. Gupta, and M. P. A. Fisher, Phys. Rev. Lett. **63**, 1511 (1989).

[6] M. Honda, *The Impedance Measurement Handbook, a Guide to Measurement Technology and Techniques* (Yokogawa-Hewlett-Packard LTD., 1989), p. 2-2.

[7] A. T. Dorsey, Phys. Rev. B **43**, 7575 (1991).

[8] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6146 (1993).

Chapter 6 : AC Magnetic Susceptibility

Although considerable experimental and theoretical effort has been invested in using the magnetic susceptibility (χ) technique to study the vortex properties of high temperature superconductors,[1-6] most of the research to date has focused on derivations of pinning related physical quantities such as the pinning potential and "irreversible line"[1-4] discussed in Chapter 2. Analysis of the magnetic susceptibility based on concepts of flux creep, flux flow, and diffusion, while capable of describing qualitative temperature and magnetic field dependence, has not yet provided a quantitative account for the frequency dependence[4], nor has it been reconciled with experimental evidence of a second-order vortex-glass transition.[7-10] By developing a new technique to study the frequency dependence of the magnetic susceptibility, we apply the concepts of critical scaling in this chapter to show that measurements of the ac magnetic susceptibility on high temperature superconductors are consistent with a second-order vortex-glass transition.

6.1 Conventional Techniques

There have previously been two common methods for performing magnetic susceptibility measurements: the standard two-coil technique[11] shown in Figure 6.1, and the single coil technique.[2] We first consider the standard

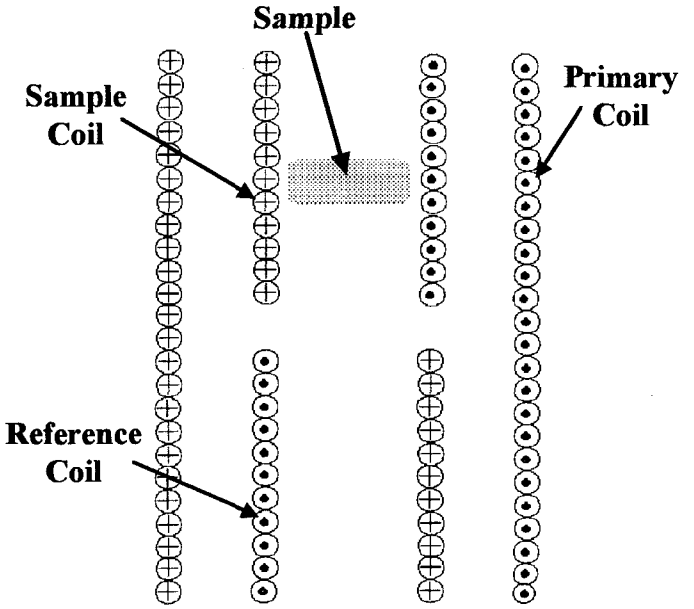


Figure 6.1. Simplified diagram of a standard two-coil susceptibility measurement system consisting of an outer primary coil and an inner sample coil.[11] The additional reference coil is added to improve the measurement resolution by subtracting the background signal V_{back} defined in Eq. (6.1).

two-coil system and use it as an example of how susceptibility is measured. The standard two-coil technique has a primary coil that provides a uniform magnetic field across the sample and a sample coil that acts as a detector to measure the magnetization induced in the sample by the applied field. More precisely, the primary coil is driven by an ac current $I(t) = I_0 \, e^{i2\pi ft}$, which leads to an excitation field $h(t) = h_0 \, e^{i2\pi ft}$ inside the sample coil. The sample responds with a magnetization M, which in the linear response regime is proportional to the applied field $(M \propto h)$. For the applied excitation field $h(t) = h_0 \, e^{i2\pi ft}$, the voltage on the sample coil is given by

$$V_{coil} = i2\pi f n \mu_0 \left[M_{eff} + \int_{area} h \cdot dA \right] \equiv V_M + V_{back} \tag{6.1}$$

where n is the number of turns on the coil, the integral is over the cross section of the coil, $M_{eff} \propto M$ is an effective magnetization of the sample, and V_M and V_{back} are the resulting magnetization signal and background signal, respectively. The exact relationship between M and M_{eff} (and therefore between M and V_M) depends on the geometry of the sample and the effectiveness of the coupling between M and the sample coil. The coupling between the sample and detector is treated empirically as a mutual inductance between the sample and sample coil in the "one-loop model" described in Chapter 7.3, but to better understand the susceptibility system, Figure 6.2 provides a more helpful illustration. The magnetization

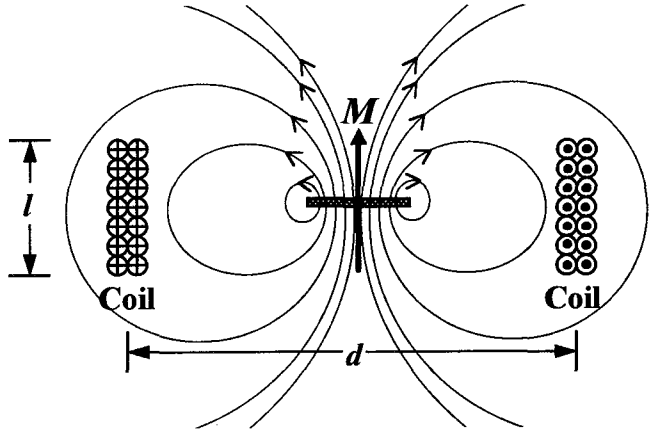


Figure 6.2. Sample magnetization M approximated as a magnetic dipole to illustrate the coupling between the sample and the coil. The applied ac field ($h \| M$) that generates M is not shown.

signal measured by the coil will be given by $V_M = i2\pi f\mu_0 \sum_{turns} \int_{area} M \cdot dA$ where the summation is over all turns of the coil, and the integral is over the plane of each turn. As is illustrated in Figure 6.2, if the coil diameter d is made too large, there will be a reduction in signal due to the flux crossing back down through the coil which reduces the net sample flux through the coil. Also, if l is too large, a smaller fraction of the sample flux will pass through the turns that are far above or below the sample. Additionally, from Eq. (6.1), the background signal is given by $V_{back} = i2\pi f\mu_0 n \int_{area} h \cdot dA$, where again, the integral is over the plane of the coil. Because h is roughly constant across the diameter of the coil (see Figure 6.5), the background V_{back} increases with increasing coil size, which, combined with decreasing sensitivity to measuring V_M, will rapidly reduce the signal to noise ratio with increasing coil size. For these reasons, it is important to have the detection coil as close to the sample size as possible in order to maximize the detected signal. Once the sample and coil geometry are established, the detected magnetization signal is $V_M \equiv i2\pi f n\mu_0 M_{eff} \propto i2\pi f n\mu_0 M$, verifying Eq. (6.1) with $M_{eff} \propto M$.

The sample susceptibility signal χ is given by $\chi \equiv \mathrm{d}M/\mathrm{d}H$, but in the linear response regime, it can be written as

$$\chi \equiv \chi' + i\chi'' \equiv \frac{M}{h} \propto \frac{V_M}{if\mu_0 hA} \propto \frac{V_{coil} - V_{back}}{if\mu_0 hA} \tag{6.2}$$

where A is the area of a coil loop. The measured χ is actually an effective sample susceptibility rather than the intrinsic susceptibility χ_m because h is the applied field, not the true local magnetic field h_L. The difference between χ and χ_m is entirely a consequence of the sample geometry and can be compensated by the demagnetization factor N.[11] The demagnetization factor is defined by $h_L = h - NM$, with the material susceptibility given by $M=\chi_m h_L$. Therefore, the material susceptibility is given by

$$\chi_m = \frac{\chi}{1 - N\chi}. \tag{6.3}$$

If the sample is approximated as an ellipsoid of revolution (a spheroid), the demagnetization factor is given by[11,12] $N = \left(1-\gamma^2\right)^{-1}\left[1-\gamma\left(1-\gamma^2\right)^{-1/2}\cos^{-1}\gamma\right]$ where γ is the ratio of the polar axis ($\approx 25\mu$m for the sample used) to the equatorial axis ($\approx$2mm), which gives $\gamma \approx 0.012$ and $N \approx 0.98$. In the presence of very small magnetic fields (<~100 Oe), the Meissner state with $\chi_m \approx -1$ is found in $YBa_2Cu_3O_7$ crystals[13,14], which as shown in Figure 6.3 makes the demagnetization factor very important. However, for experiments performed in large dc magnetic fields, the sample is in the mixed state with flux lines passing through the sample which, along with the large magnetic penetration depth, is found to drive the superconductor far from the Meissner state[13,14] so that $h \approx h_L$ and $|\chi_m| << 1$. Measurements on a $YBa_2Cu_3O_7$ crystal have shown that with an applied dc field of 3kOe and a temperature $T = 89.5\text{K}$, the actual susceptibility is only[13] $|\chi_m| \approx 0.03$. The susceptibility decreases in the presence of larger magnetic fields, so all of the data presented in section 6.3 for magnetic fields from 3kOe to 25kOe satisfies $|\chi_m| \le 0.03$. As shown in Figure 6.3, for $|\chi_m| \le 0.03$, the demagnetization effect is small so that $\chi \approx \chi_m$.

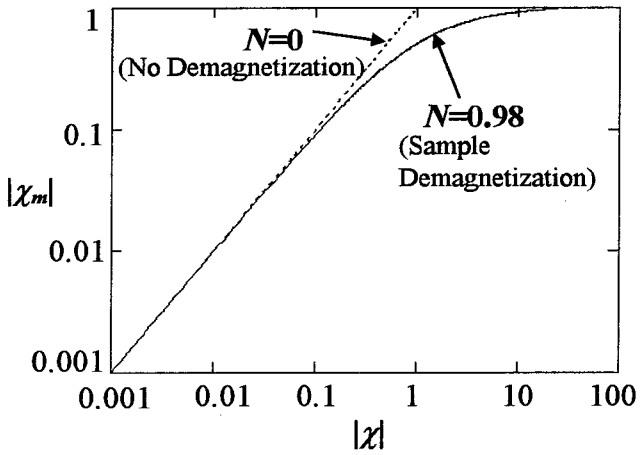


Figure 6.3. The effects of the demagnetization factor on the susceptibility. The sample susceptibility is χ, and the material susceptibility after correcting for demagnetization effects is χ_m. The solid line uses the estimated demagnetization factor for the sample, and the dashed line is with no demagnetization effects ($\chi=\chi_m$).

One of the most serious problems with the standard two-coil design is that the sample magnetization signal is much smaller than the background coil signal, $V_M << V_{back}$

(see Eq. (6.1)), leading to a small signal to noise ratio. A common solution to the signal to noise problem is to add the reference coil shown in Figure 6.1, with windings in the opposite direction of the sample coil. With the reference coil and sample coil connected in series, the measured voltage will be $V=V_{coil}+V_{comp}$, where $V_{comp}=-V_{back}$ is the signal from the empty reference coil. The measured signal is thus given by $V=(V_M+V_{back})+V_{comp}=V_M$, which is exactly the desired signal. Although this concentric coil design works very well at low frequencies, there is generally a large capacitive coupling between the primary coil and sample coil. At frequencies $f>\sim 10^4$ Hz[11] the capacitive coupling will dominate the inductive coupling that is responsible for detecting the magnetization M, making such a design unusable for the broad frequency range desired in these measurements. One alternative design that works well at higher frequency is to use a single coil that acts as both the primary and sample coils.[2] A constant ac current is passed through the coil, and V_{coil} is the voltage measured across the same coil. This technique eliminates the problem of capacitive coupling between coils and has been used to measure the susceptibility at frequencies up to ~2MHz[2]. However, the single coil approach provides no way to directly subtract the background signal V_{back}, thus substantially reducing the signal to noise ratio. Some preliminary measurements with a single coil design showed that $V_{back} \approx 95\% \times V_{coil}$, making small changes in the sample susceptibility unmeasurable.

6.2 Improved Design

In order to make sensitive susceptibility measurements over a broad frequency range, the advantages of the standard two-coil technique and the single coil technique are combined into a differential two-coil measurement technique. The differential two-coil setup is shown in Figure 6.4 and uses two nearly identical coils placed approximately 2cm apart so as to eliminate any detectable coupling. The sample is placed in one of the coils

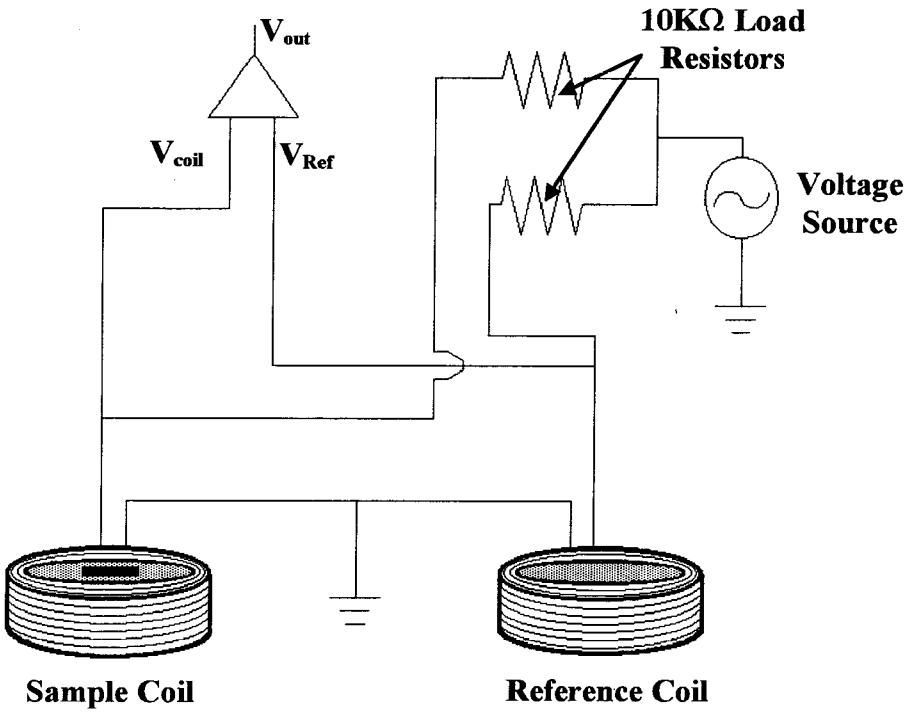


Figure 6.4. Schematic drawing of the differential two-coil susceptibility measurement system.

and the other is left empty to serve as a reference coil. By using a differential amplifier, the two signals are directly subtracted to give an output voltage $V_{out} = V_{coil} - V_{ref}$ as shown in Figure 6.4. The calculations from the standard two-coil method remain valid for the differential two-coil configuration, so from Eq. (6.1) $V_{coil}=V_M+V_{back}$. Because the reference coil is nearly identical to the sample coil (except that it is empty), we have $V_{ref}=V_{back}$. Combining this approximation with Eq. (6.2) yields $V_{out} = V_{coil} - V_{ref} = V_M \propto \chi \cdot (if)$ for a constant amplitude h of the applied ac field. Unfortunately, in practice isolating the true signal is not quite so easy.

The coils are made with 24 turns of 0.002 inch diameter copper wire held together with GE varnish. The coil has an inductance of 10μH and with an applied current of 25μA generates a central field of 4.3×10^{-3}Oe as shown in Figure 6.5. The sample

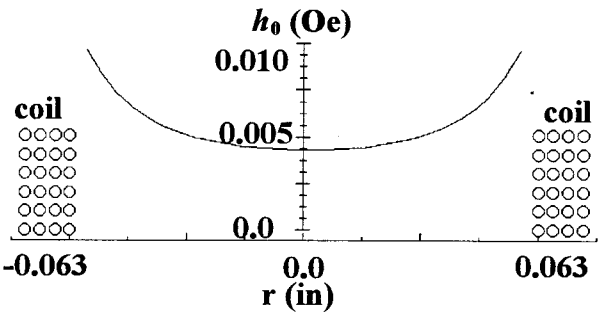


Figure 6.5. Profile of magnetic field generated by the coil. Here r is the radial distance from the coil center, and the coil cross section is shown with an inner diameter of 0.125 inches.

used in this work extends across r≈±0.04 inches, which is just inside the sharp upturn in the applied field h_0(r). Although a larger coil would produce a more uniform field across the sample, as discussed above, a larger sample coil leads to a much smaller signal to noise ratio. The coil diameter is chosen as a reasonable compromise between signal level and field uniformity. Actually, it is not expected that the magnetization signal would be distorted by a non-uniform h_0 provided h_0 is small enough that the sample remains in the linear response regime, but a more uniform h_0 is still preferable. The two coils used for the sample and reference coils vary by ≈0.5% in inductance and resistance, which produces an additional background signal ≈10% × the sample signal. Additional contributions to the background as well as amplitude and phase distortions are caused by the wiring and electronics, but all of these effects can be compensated with an accurate calibration without adversely affecting the measured signal.

The primary instrument used in the susceptibility measurements is an HP4194A impedance analyzer as shown in Figure 6.6. The HP supplies a 0.5V driving signal to the coils. A 10kΩ non-inductive load resistor is placed in series with each of the coils to restrict the current to 25μA. The coil impedance is primarily inductive so that $|Z_{coil}|=2\pi fL$ where L≈10μH is the coil inductance. This impedance Z_{coil} is largest at high frequency and reaches $|Z_{coil}|$≈20Ω at f=2MHz, the highest frequency used. Because $|Z_{coil}|$<<10kΩ (the resistance of the load resistor), the current is limited almost exclusively by the load resistor so any small changes in the coil impedance will not significantly change the current through each coil. More importantly, the large load resistors will also ensure that the current is the same through both coils, with the total current monitored by the HP. The voltages of the two coils are measured relative to a common ground between the coils, and these voltages are subtracted and amplified by a 62dB gain low noise preamplifier. The output of the amplifier is then read by the impedance analyzer, and upon dividing the

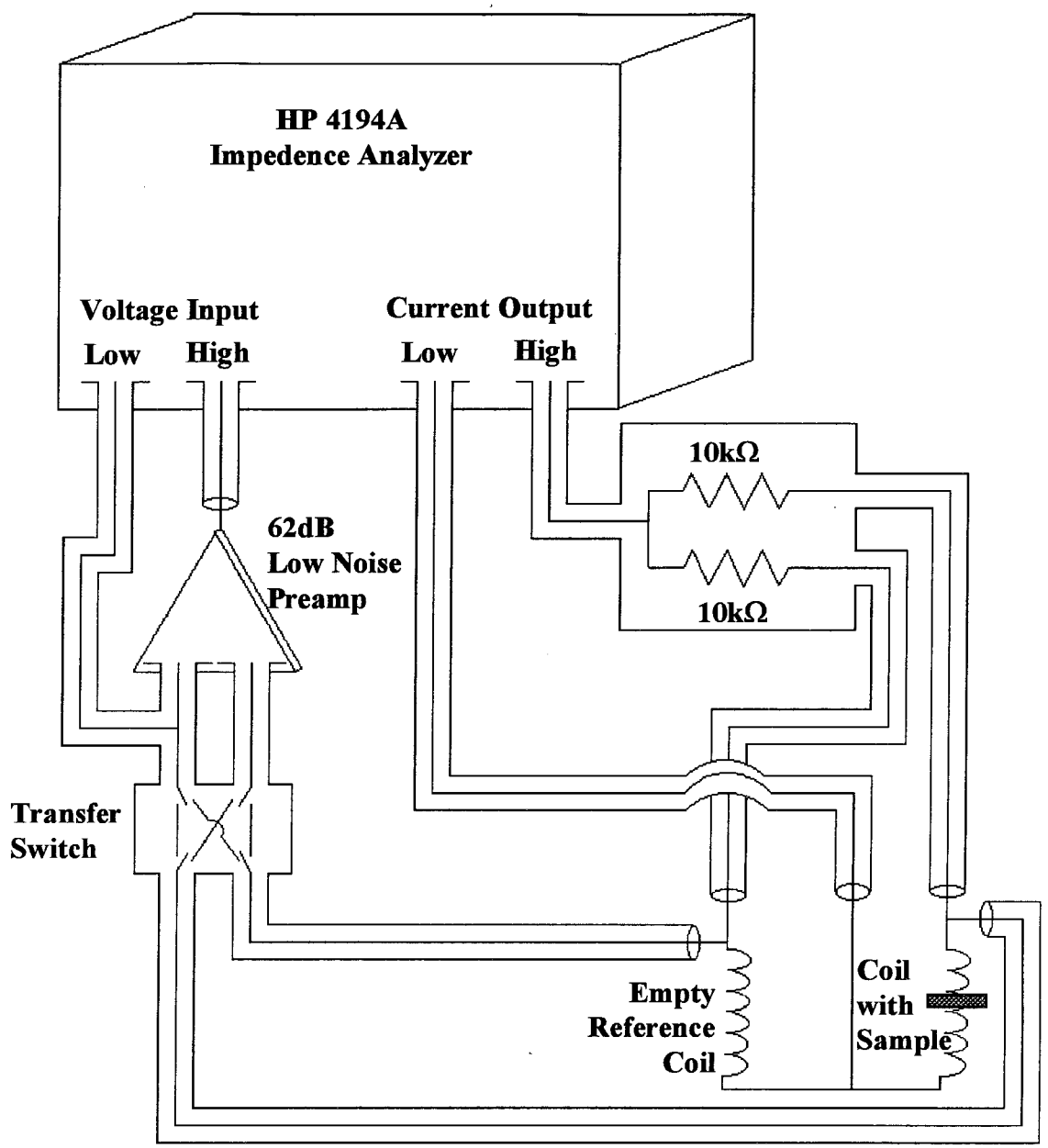


Figure 6.6. Block diagram of ac magnetic susceptibility measurement system. A more accurate drawing of the measurement coils is shown in Figure 6.4.

output voltage by the output current of the HP, the result is interpreted as an effective impedance $Z_{meas} \equiv {}^{V_{out}}\!/_{I_{out}}$. More accurately, Z_{meas} is the difference in impedance between the coil with the sample and the empty reference coil, $Z_{meas} = Z_{coil} - Z_{ref}$, so from Eq. (6.2), the susceptibility should be given by $Z_{meas} \propto i\chi f$. However, there is an ampli-

tude distortion $G(f)$, and phase shift $\theta(f)$ due to wiring and electronics, so the measured impedance is more accurately given by

$$Z_{meas} = \left[(i\chi f)G(f)e^{i\theta(f)} + Z_{back}(f)\right] \tag{6.4}$$

where Z_{back} accounts for both a stray inductance and the slight mismatch in the coils. Finding the calibration functions, $G(f)$, $\theta(f)$, and $Z_{back}(f)$, requires a detailed frequency dependent calibration. For temperatures well above the superconducting transition, the susceptibilty of the superconductor becomes zero, $\chi(T >> T_M) \rightarrow 0$, so the measured signal is just the background, $Z_{meas}(T >> T_M) \approx Z_{back}$. Typically, the background is only $Z_{back} \approx 10\% \times Z_{meas}$, so it does not significantly reduce the signal to noise ratio. The background signal does show some temperature dependence over a broad temperature range such as 300K to 70K, but for the temperature intervals of interest in the measurements (typically ~2K), this temperature dependence is below the experimental resolution. However, the temperature dependence in Z_{back} requires that it be measured at a temperature that is not too far above the transition temperature, typically at ≈95K for T_c≈92K. The amplitude and phase distortion ($G(f)$,and $\theta(f)$) are due primarily to wiring and the electronic components outside the dewar and therefore show no significant temperature dependence. In order to calibrate these distortions, a reference inductor with impedance Z_{ind} is attached in parallel with the empty sample coil giving an impedance $Z_{cal} = \left(Z_{coil}^{-1} + Z_{ind}^{-1}\right)^{-1}$. The impedance of the sample coil, Z_{coil}, and the impedance of the combined coil and inductor, Z_{cal}, can be accurately measured by using the HP impedance analyzer in a separate probe calibrated with factory calibration standards. With the combined coil and inductor placed back into the sample probe, the new signal measured with the differential two-coil technique is given by

$$Z'_{meas} = \left[(Z_{cal} - Z_{coil})G(f)e^{i\theta(f)} + Z'_{back}\right]. \tag{6.5}$$

The gain calibration is done at room temperature because Z_{cal} can only be measured accurately at room temperature, so Z'_{back} is the background measured at room temperature. The background Z'_{back} is measured in the same way as Z_{back} described above with only the two coils in the sample probe (no reference inductor) except that Z'_{back} is measured at room temperature. Since (Z_{cal} - Z_{coil}) is known, the distortion is given by $G(f)e^{i\theta(f)} = (Z'_{meas} - Z'_{back})/(Z_{cal} - Z_{coil})$. Because the $G(f)$ and $\theta(f)$ are due primarily to cabling and the electronics outside the dewar, $G(f)e^{i\theta(f)}$ is temperature independent so the room temperature calibration is all that is needed. Thus we find $\chi(f) \propto \frac{-i}{f}[Z_{meas} - Z_{back}]/[G(f)e^{i\theta(f)}]$ where the unknown proportionality constants that remain are from the sample volume and coupling between the sample and the coil. Although these constants could be determined by measuring a material with a known susceptibility and of the same size and shape as the sample, the absolute value of the susceptibilty is not important to the analysis presented in this chapter. Instead, we simply normalize the susceptibility in such a way that the real part is $\chi'(T >> T_c) = 0$ when the superconductor is in the normal state, and $\chi'(T << T_c) = -1$ at temperatures well below the superconducting transition. The normalization is not meant to imply that the sample is fully in the Meissner state (actually, $|\chi| \leq 0.03$ as explained on p. 6-4), but merely represents the choice of arbitrary units with the maximum susceptibility amplitude defined to be $|\chi|_{max} = 1$.

6.3 Experimental Results

The data are taken using a procedure similar to the one used for performing frequency dependent impedance measurements. The temperature is held fixed to within ±0.003K, and the susceptibility is measured at 350 frequencies for 10^3 Hz to 2×10^6 Hz. The transfer switch then reverses the voltage leads and the measurements are repeated for

all frequencies. As discussed in Chapter 5, the transfer switch is used to correct a problem found internal to the HP4194A, and other than correcting this problem, the switch should have no effect on the measured data. The complete frequency scan requires the temperature to be stable for ≈20 minutes. The temperature setpoint is then increased by ≈0.03K, and after the temperature is stable (usually 5~10 min.), the frequency scan is repeated. Using this procedure, a series of χ versus f isotherms is taken as shown in Figure 6.7 and Figure 6.8. The series of isotherms spanning the phase transition is taken in various dc magnetic fields from 3 to 25 kOe.

The data presented here are composed of measurements made in six different magnetic fields from 3 to 25 kOe with 6 and 25 kOe selected as representative of all fields. Figure 6.7(a) and Figure 6.7(b) show the susceptibility amplitude $|\chi|$ versus frequency isothermal curves, and the corresponding phase (π-ϕ_χ) of the magnetic susceptibility, defined by $\chi = |\chi| e^{i(\pi-\phi_\chi)}$, is shown in Figure 6.8. This definition of the phase is chosen so that $0 < \phi_\chi < \pi/2$.

In order to use these data to investigate the vortex-solid to vortex-liquid phase transition, we use the scaling analysis described in Chapter 3. There, we derived the following scaling relations for the amplitude and phase of the susceptibility:

$$\begin{aligned} \tilde{\chi}_{\pm}\left(\tilde{f}\right) &\equiv |\chi| \left|1-\left(T/T_M\right)\right|^{-\nu/2} \\ \tilde{\phi}_{\chi\pm}\left(\tilde{f}\right) &\equiv \phi_\chi \\ \tilde{f} &\equiv f \left|1-\left(T/T_M\right)\right|^{-\nu z} \end{aligned} \tag{6.6}$$

with $\tilde{\chi}_{\pm}$ and $\tilde{\phi}_{\chi\pm}$ being the universal scaling functions for the amplitude and phase, and $T>T_M$ (+) and $T<T_M$ (-). As discussed in Chapter 3, Eq. (6.6) requires that for $T \to T_M$ the susceptibility has a positive power law frequency dependence $|\chi| \to f^{1/(2z)}$, which we verify is consistent with the experimental data shown in Figure 6.7. Furthermore, Eq. (6.6)

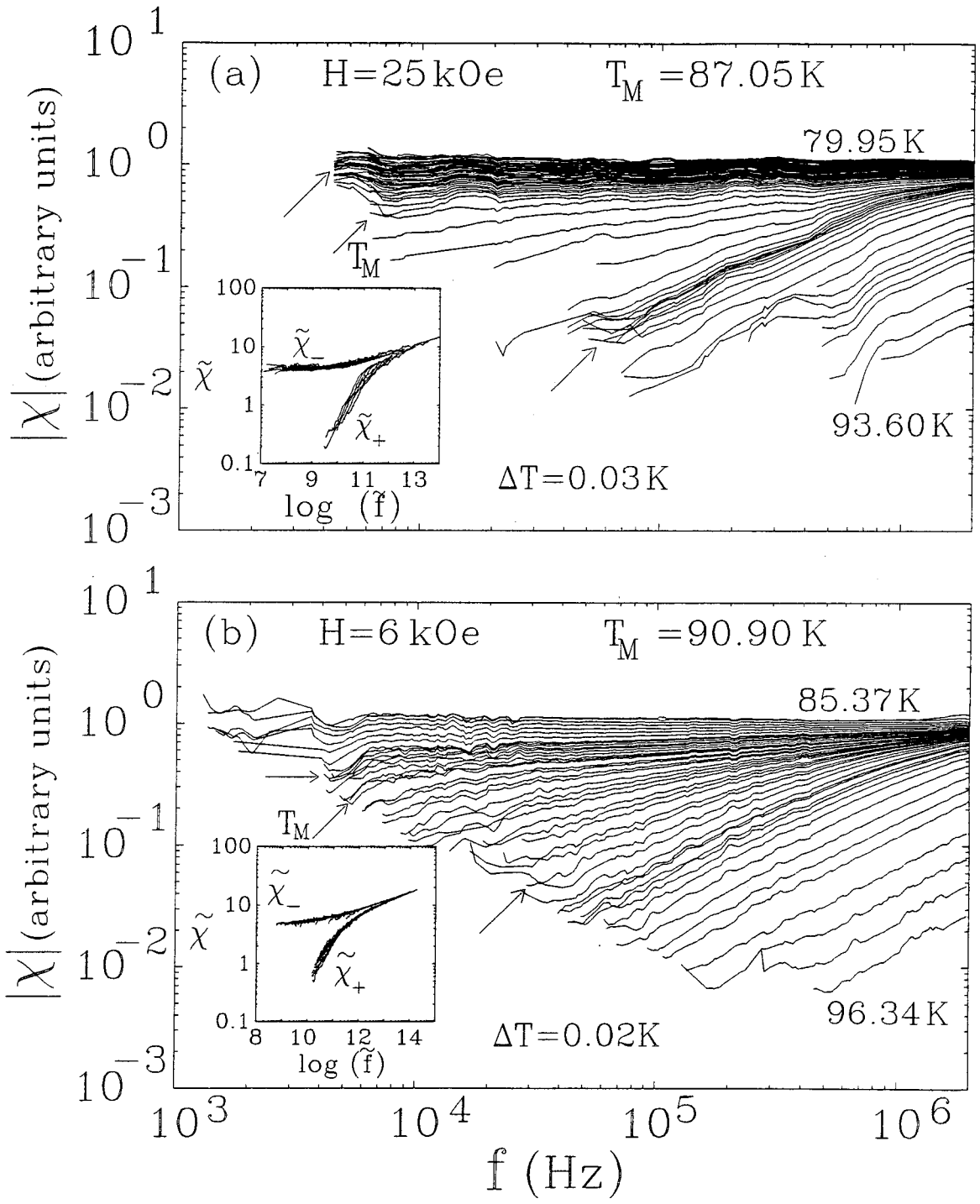


Figure 6.7. Susceptibility amplitude $|\chi|$ vs. frequency (f) isotherms for (a) H=25kOe and (b) H=6kOe. The arrows indicate the isotherms in the critical regime that are scaled into universal functions $|\tilde{\chi}_{\pm}(\tilde{f})|$ shown in the insets. The total temperature ranges and the temperature increment ΔT in the critical regime is given in the figures.

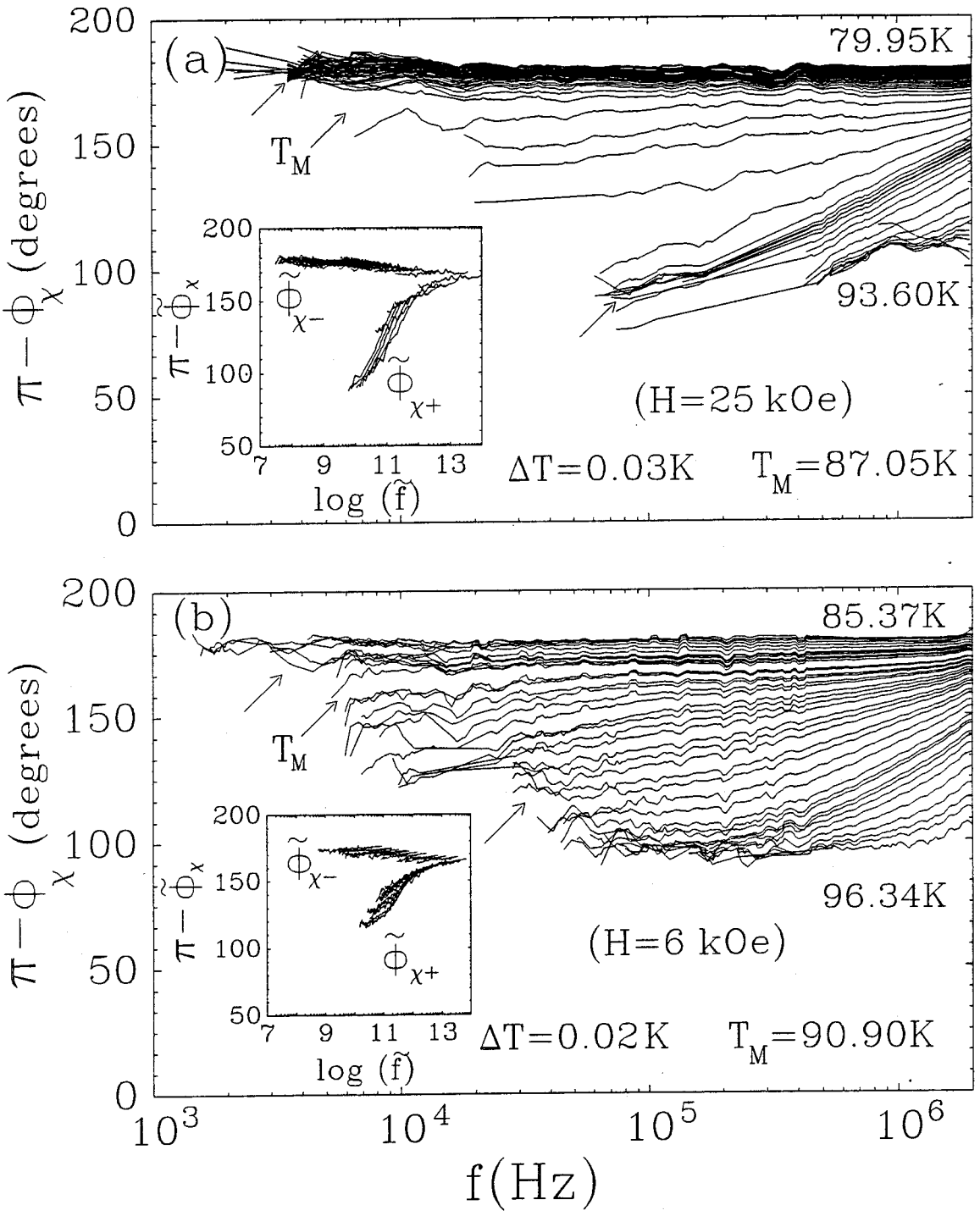


Figure 6.8. Susceptibility phase (π-ϕ_χ) vs. frequency (f) isotherms for (a) H=25kOe and (b) H=6kOe. The arrows indicate the isotherms that are scaled into the universal functions $\left[\pi - \phi_{\chi\pm}(\tilde{f})\right]$ in the insets. The total temperature ranges and the temperature increment ΔT in the critical regime are as shown.

ensures that $|\chi(f \to 0)| \sim |1-(T/T_M)|^{\nu/2} \to 0$ as $T \to T_M$, consistent with the data in Figure 6.7 and Figure 6.9. These points are in sharp contrast to an incorrect previous derivation of the susceptibility scaling relation[15] $\chi \sim \xi$ discussed in Chapter 2, which implies that for $T \to T_M$, $\chi(f)$ has a negative power law, $|\chi| \to f^{-1/z}$, and $|\chi(f \to 0)| \sim |1-(T/T_M)|^{-\nu} \to \infty$, contradicting the experimental data shown in Figure 6.7 and Figure 6.9.

Using Eq. (6.6), the χ versus f isotherms shown in Figure 6.7 can be "collapsed" into a single set of universal functions $\tilde{\chi}_{\pm}(\tilde{f})$ shown in the insets by using the exponents z=3.0 ± 0.3 and ν=0.70 ± 0.05. The same exponents are used for all magnetic fields, and the resulting universal functions $\tilde{\chi}_{\pm}(\tilde{f})$ are found to be the same for all fields after accounting for the field dependent constant factor. Scaling analysis of the phase provides another independent test for universality, and the insets of Figure 6.8 show that within

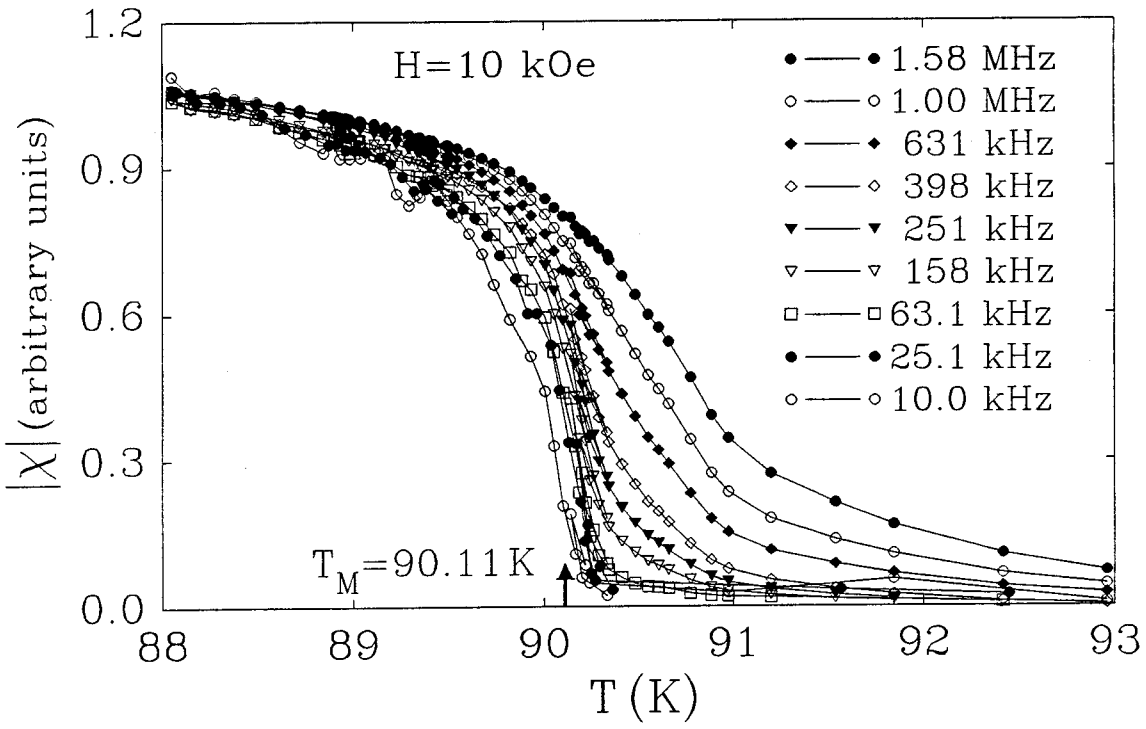


Figure 6.9. Susceptibility amplitude $|\chi|$ vs. temperature (T) for H=10kOe, shown for several different frequencies. Note that the low frequency susceptibility approaches 0 smoothly at the vortex-glass transition T_M despite the presence of a peak in the imaginary component.

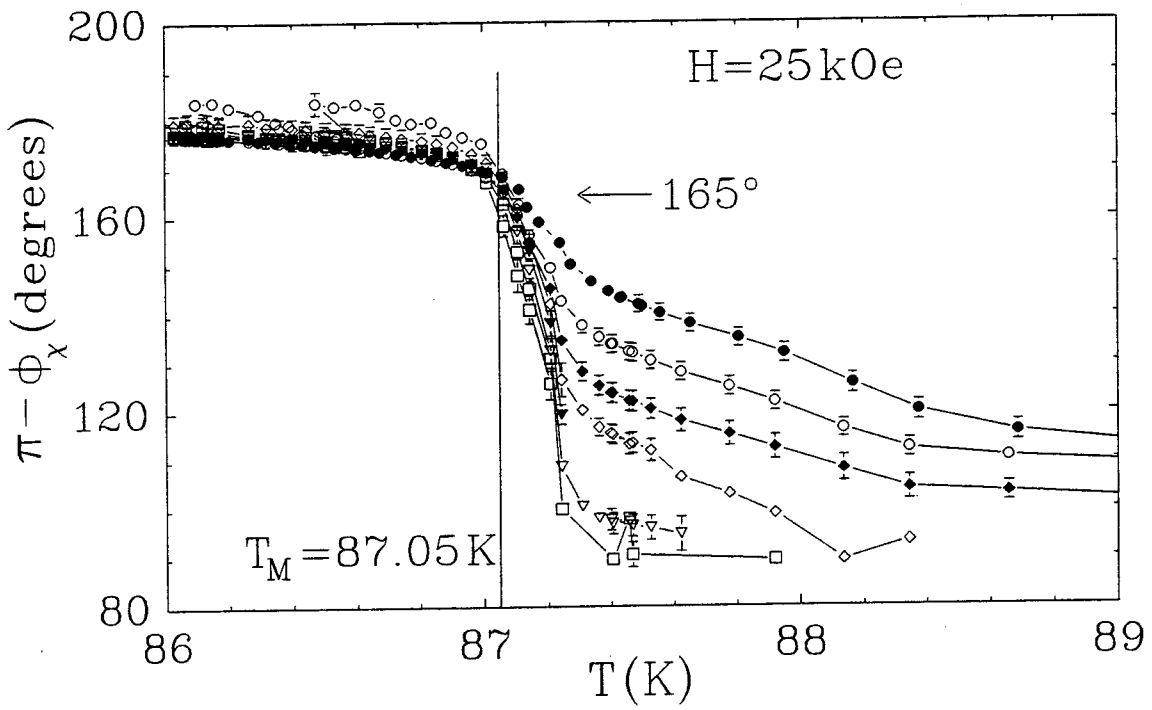


Figure 6.10. Susceptibility phase (π–ϕ_χ) vs. temperature (T) for H=25kOe at several different frequencies. At the transition temperature T_M, the phase approaches a nearly frequency independent value of 165° ± 5°, which corresponds to $z \approx 3.0$.

experimental accuracy the same exponents (z=3.0 ± 0.3, and ν=0.70 ± 0.05) and transition temperatures $T_M(H)$ derived from the amplitude data also yield universal functions $\left[\pi - \tilde{\phi}_{\chi\pm}(\tilde{f})\right]$ for the phase. In addition, by using the Kramers-Kronig relation, [10,16] so that

$$\pi - \phi_\chi = -\mathcal{P}\int_{-\infty}^{\infty} \frac{\omega'}{\pi}\frac{\ln|\chi(\omega')|}{\omega' - \omega}, \qquad \pi - \phi_\chi(T_M) = \pi - \frac{\pi}{2}\frac{1}{2z} \tag{6.7}$$

where $\mathcal{P}$ denotes the principal value of the integration, we find that the phase is frequency independent at the transition temperature. By transforming the data in Figure 6.8 to the phase versus temperature data shown in Figure 6.10, the phase at the transition temperature T_M is found to be 165° ± 5°. From Eq. (6.7), this value of $\phi_\chi(T_M)$ gives a value of z=3.0 ± 0.3 which is again consistent with the value found by "collapsing" the χ vs. f isotherms. The self consistency of these critical exponents and universal functions in

various magnetic fields and two independent parameters (χ and ϕ_χ) is essential to verifying the universality that is associated with a second-order phase transition. Additionally, these critical exponents are in excellent agreement with those obtained from our previous dc and ac transport measurements[7-10] which verifies that the same vortex-solid to vortex-liquid phase transition is found with three different experimental techniques used to probe the vortex properties.

6.4 Discussion

Having demonstrated the validity of critical scaling of χ near the vortex-glass transition, it is useful to compare our data with the conventional electrodynamical model for the magnetic susceptibility.[1,5,6] By converting the amplitude and phase versus frequency isotherms in Figure 6.7 and Figure 6.8 to $\mathrm{Im}(\chi) \equiv |\chi| \sin(\pi - \phi_\chi)$ versus temperature data at fixed frequency, one obtains the familiar series of "peak" features shown in Figure 6.11(a) for different magnetic fields. The positions of these peaks are often used to identify the "irreversible line," $T_{irr}(f, H)$, in the vortex phase diagram.[2-4] As shown in Figure 6.11(b), these peak positions are frequency dependent with only the lowest frequency peaks corresponding to the temperature of the true phase transition (T_M) found from scaling analysis.

To understand the susceptibilty better, the analytical expression for a slab of thickness w yields[1,5,6]

$$1+\chi = (1+\chi') + i\chi'' = \left(2\tilde{\lambda}/w\right) \tanh\left(w/2\tilde{\lambda}\right) \tag{6.8}$$

where $\tilde{\lambda}$ is the complex ac penetration depth given by[6]

$$\tilde{\lambda}^2 = \left(\lambda_R^2 - \lambda_I^2\right) + i\left(2\lambda_R\lambda_I\right) = \frac{\lambda^2 + i\left(\tilde{\delta}_{vc}^2/2\right)}{1 - 2i\left(\lambda^2/\delta_{nf}^2\right)}. \tag{6.9}$$

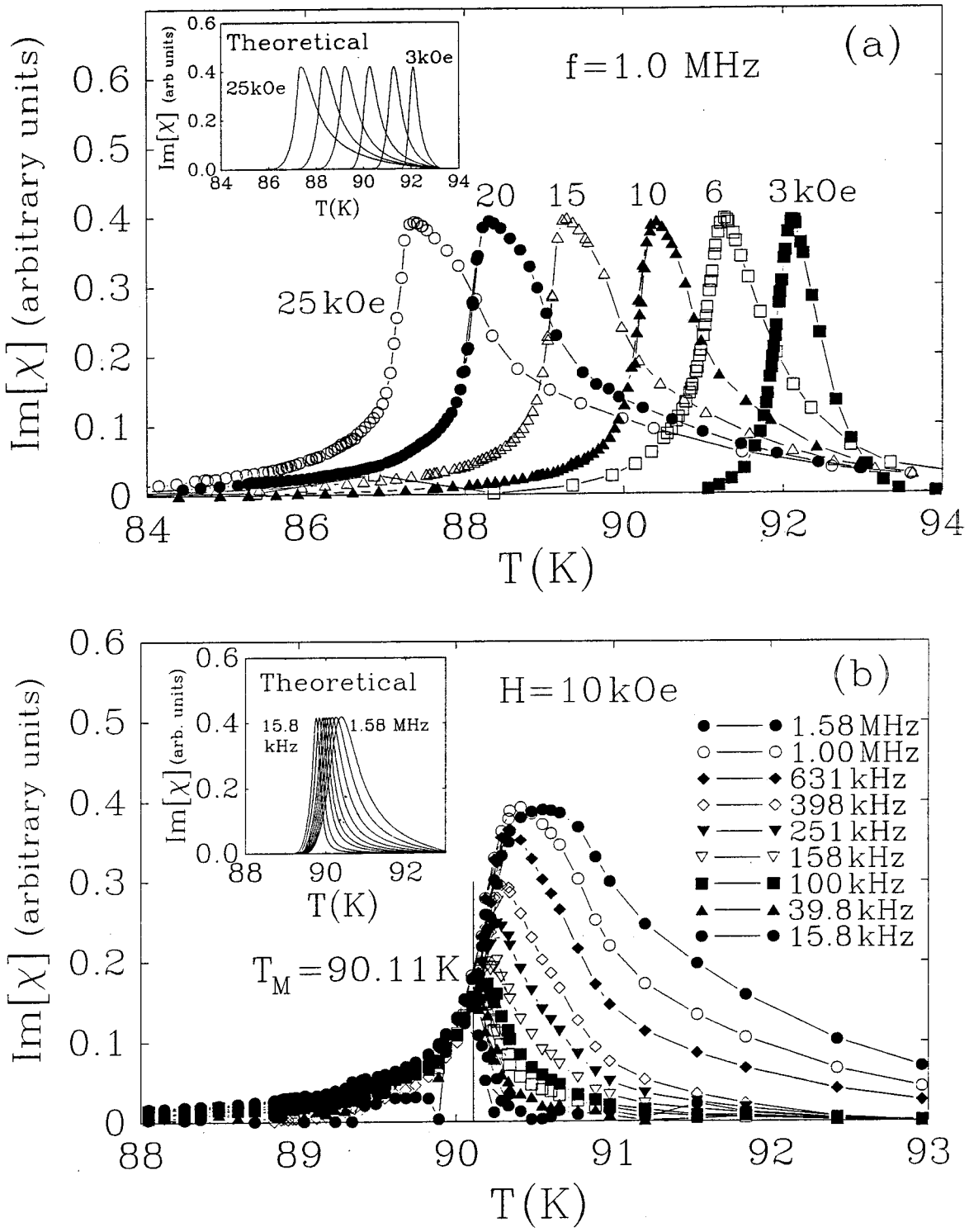


Figure 6.11. Imaginary part of the susceptibility Im[χ] shown for (a) f=1.0 MHz at different fields, and (b) H=10kOe at different frequencies. The insets show the corresponding theoretical curves obtained by using Eq. (6.10).

In Eq. (6.9), $\tilde{\delta}_{vc}$ denotes a complex skin depth due to thermally induced vortex motion[6] and δ_{nf} is the normal fluid skin depth.[6] Thus, the imaginary part of the susceptibility can be written as

$$\chi'' = \frac{2}{w}\left[\lambda_I \sinh\left(\frac{\lambda_R w}{|\tilde{\lambda}|^2}\right) - \lambda_R \sinh\left(\frac{\lambda_I w}{|\tilde{\lambda}|^2}\right)\right]\left[\cosh\left(\frac{\lambda_R w}{|\tilde{\lambda}|^2}\right) - \cosh\left(\frac{\lambda_I w}{|\tilde{\lambda}|^2}\right)\right]^{-1}. \qquad (6.10)$$

The skin depths δ_{nf} and $\tilde{\delta}_{vc}$ involve various material parameters, and the theoretical curves shown in the insets of Figure 6.11 use the following temperature and magnetic field dependencies: the pinning force constant $k_p(T) = k_p(0)\left|1-(T/T_c)^2\right|^2$, and the effective viscosity $\eta(T) = \left[H_{c2}(T)\Phi_0/\rho(T)\right]$ where Φ_0 is the flux quantum and $\rho(T)$ is taken directly from the experimental data in Figure 1.4(a). Additional parameters are the upper critical field $H_{c2}(T) = H_{c2}(0)\left|1-(T/T_c)^2\right|$, the penetration depth[6,17] $\lambda_L(T) = \lambda_L(0)\left|1-(T/T_c)^2\right|^{-1/2}\left|1-(H/H_{c2})\right|^{-1/2}$, using numerical factors $\lambda_L(0) = 1400\,\text{Å}$, $H_{c2} = 200\,\text{T}$, and $k_p(0) = 10^6\,\text{N/m}^2$. Conventional pinning theory[1,5,6] asserts a finite resistivity except for T=0, giving rise to a continuously decreasing $T_{irr}(f,H)$ with decreasing f, which is inconsistent with the experimental frequency dependence shown in Figure 6.11. On the other hand, by considering the measured linear resistivity that drops to $\rho(T < T_M(H)) = 0$ for temperatures below the melting transition, we find reasonable agreement with the experimental Im[χ] vs. T data as shown in the insets of Figure 6.11. The field dependence of Im[χ] vs. T data is accurately reflected with the theoretical model expressed in Eq. (6.10). Similarly, the frequency dependence of Im[χ] vs. T in the inset of Figure 6.11(b) also shows that the peak positions converge to the true phase transition temperature with decreasing frequency, although the peak heights are not accurately represented in this model. We note that the analytic form in Eq. (6.10) involves many parameters such as the viscosity $\eta(H,T)$ and force constant $k_p(H,T)$ which have field and

temperature dependencies that are not well established. Because these parameters are not well known, inserting the correct resistivity alone may not be sufficient to account for all details of the frequency dependence. Finally, we notice that $T_{irr} \rightarrow T_M$ for $f <\sim 100\,\text{kHz}$ as depicted by the phase diagram in Figure 6.12.

This work has investigated the critical scaling of the frequency dependent magnetic susceptibility near the vortex-glass transition in $YBa_2Cu_3O_7$ single crystals. In the linear response regime, the susceptibility χ follows the scaling relation $\chi \sim |1 - T/T_M|^{\nu/2}$ with critical exponents ν=0.75±0.05 and z=3.0±0.3. These exponents and the transition temperature $T_M(H)$ are consistent with previous values found from ac and dc transport measurements, showing conclusive evidence for the existence of a second-order vortex-solid to vortex-liquid phase transition. The results shown in Figure 6.11 and Figure 6.12 also

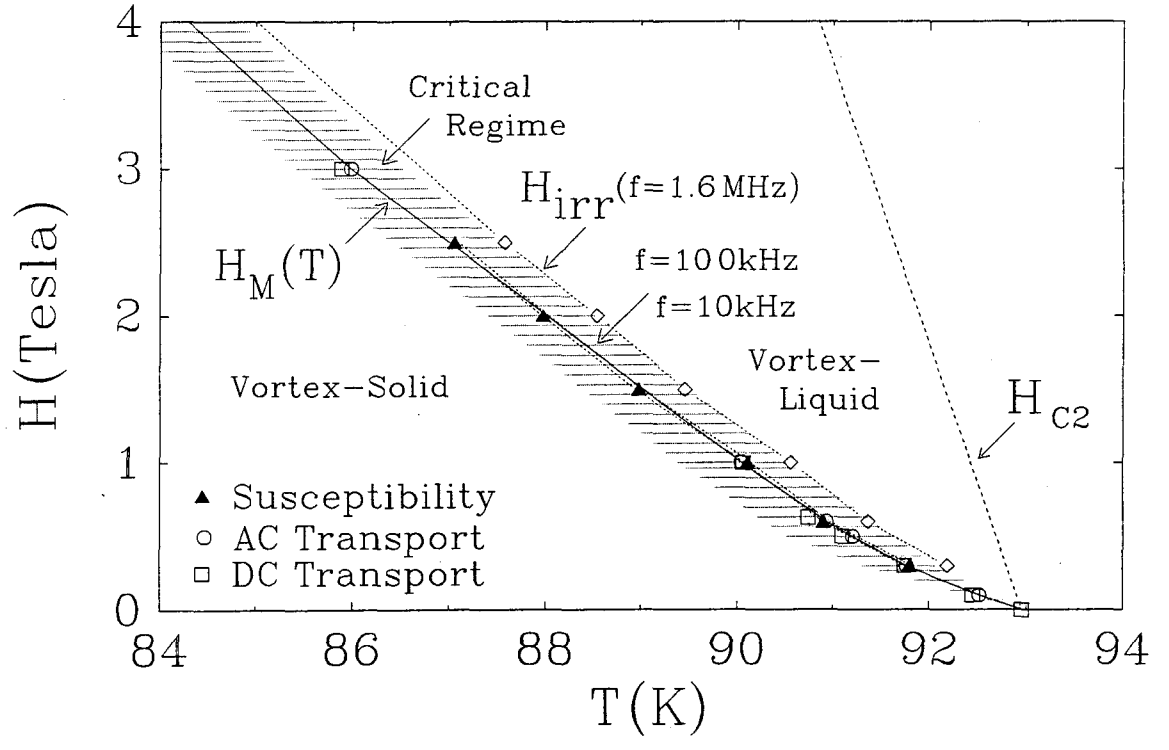


Figure 6.12. Vortex phase diagram for a $YBa_2Cu_3O_7$ single crystal. The vortex-glass melting line $H_M(T)$ obtained from ac magnetic susceptibility measurements is shown to be consistent with those obtained from the transport measurements. The shaded area denotes the critical regime of $H_M(T)$, and the irreversible lines $H_{irr}(f,T)$ are obtained by identifying he peak positions of the Im[χ] vs. T data.

indicate that conventional electrodynamic theory for the magnetic susceptibility in the low frequency limit can be reconciled with a vortex-glass transition, provided that the linear resistivity $\rho(T,H)$ vanishes below $T_M(H)$, as shown in Figure 1.4(a). This work is the first demonstration of magnetic susceptibility data used to rigorously investigate the second-order vortex-solid to vortex-liquid phase transition. Having established this technique on the vortex-glass system, it can also be used to investigate other second-order phase transitions in type-II superconductors.

References

[1] P. H. Kes, J. Aarts, J. Vandenberg, C. J. Van der Beek, and J. A. Mydosh., Supercond. Sci. Technol. **1**, 242 (1989).

[2] T. K. Worthington, F. Holtzberg, and C. A. Feild, Cryogenics **30**, 417 (1990).

[3] For a general review of the experimental and theoretical work on magnetic susceptibility, see for example, C. J. van der Beek, Ph.D. thesis, University of Leiden, The Netherlands, 1992, and references therein.

[4] D. G. Steel and J. M. Graybeal, Phys. Rev. B **45**, 2643 (1992).

[5] E. H. Brandt, Intl. J. Mod. Phys. B **5**, 751 (1991); Physica C **195**, 1 (1992).

[6] M. Coffey and J. R. Clem, Phys. Rev. B **45**, 9872 (1992); Phys. Rev. Lett. **67**, 386 (1991).

[7] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6150 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 169.

[8] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6146 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 413.

[9] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, D. A. Beam, M. Konczykowski, T. A. Tombrello, and F. Holtzberg, Phys. Rev. B **47**, 8308 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 789.

[10] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, F. Holtzberg, M. Konczykowski, C. C. Tsuei, and C. C. Chi, Physica A 200, 374-383 (1993).

[11] R. B. Goldfarb, M. Lelental, and C. A. Thompson, *Magnetic Susceptibility of Superconductors and Other Spin Systems*, edited by R.A. Hein, T. L. Francavilla, and D.H. Liebenberg (Plenum Press, New York, 1992).

[12] E. C. Stoner, Phil. Mag., ser. 7, **36**, 803 (1945); J. A. Osborn, Phys. Rev. **67**, 351 (1945).

[13] L. Krusin-Elbaum, A. P. Malozemoff, Y. Yeshurun, D. C. Cronemeyer, and F. Holtzberg, Physica C **153-155**, 1469 (1988).

[14] L. Krusin-Elbaum, A. P. Malozemoff, and Y. Yeshurun, *High-Temperature Superconductors*, MRS Symposia Proceedings No. 99 (Materials Research Society, Pittsburgh, 1988), p. 221.

[15] D. S. Fisher, M. P. Fisher, and D. A. Huse, Phys. Rev. B **43**, 130 (1991).

[16] A. T. Dorsey, Phys. Rev B **43**, 7575 (1991).

[17] N.-C. Yeh, U. Kriplani, W. Jiang, D. S. Reed, D. M. Strayer, J. B. Barner, B. D. Hunt, M. C. Foote, R. P. Vasquez, A. Gupta, and A. Kussmaul, Phys. Rev. B **48**, 9861 (1993).

Chapter 7 : Hall Probe Magnetometer

One of the most challenging research topics in the vortex properties of high temperature superconductors is the nonlinear vortex response.[1-7] In particular, the novel vortex phases and dynamics incurred by controlled columnar defects in high temperature superconducting single crystals[8-10] have attracted intense experimental and theoretical efforts.[3-6,11-14] Among various experimental techniques[13-18] for investigating the vortex dynamics, measurement of the third harmonic transmissivity[13-15,19] T_{H3} is recognized as a sensitive tool which provides useful information about the onset of nonlinear response. A single inductive loop model has been developed to describe the third harmonic data in the limits of ohmic response, critical state, flux flow and "flux creep" with power law current-voltage characteristics[15]. Despite qualitative agreement, these models have not self consistently accounted for the dependence of T_{H3} over a broad range of temperatures, ac excitation frequencies, as well as the dc magnetic field strength and orientation. Thus, regardless of its numerous experimental advantages, the usefulness of the third harmonic technique is undermined by the lack of quantitative theoretical understanding. We overcome these problems by presenting a new analysis technique which interprets the transmissivity in terms of a characteristic relaxation time τ and crossover current density j. Analysis based on this "τ-j" model provides quantitative account for all experimental information.

This chapter begins with a description of the Hall probe magnetometer and how it is used to measure both the fundamental transmissivity T_{H1} and third harmonic transmissivity T_{H3}. This description of the experimental setup is followed by a discussion of the "τ-j" numerical analysis technique developed based on the "one-loop" model. Measurements of the transmissivity of a superconducting $YBa_2Cu_3O_7$ single crystal with columnar defects

are presented and analyzed in terms of the "τ-j" model. The results of the numerical analysis are then compared to predictions of the Bose-glass phase transition.

7.1 Defining the Transmissivity

The physical quantity measured by the Hall probe is the magnetic flux ϕ that penetrates through the sample. From the flux, the transmissivity T_H is defined by

$$T_{Hn} \equiv T'_{Hn} - iT''_{Hn} = \frac{1}{\pi}\int_0^{2\pi} \phi e^{-in\theta} d\theta \tag{7.1}$$

where ϕ is the flux which penetrates the sample and n specifies the harmonic of the transmissivity with n = 1 being the fundamental, and $\theta = 2\pi ft$. The transmissivity is directly related to the susceptibility by $T_H = 1 + \chi$, but it is measured only as a sample property so that details of the flux penetration and demagnetization (as described in Chapter 6.1) are not considered. Only the odd harmonics of the transmissivity will be generated by the sample as can be seen by the following argument. By time reversal symmetry, for the applied field $h = h_0 \cos(\theta)$, one has $\phi(\theta) = -\phi(\theta+\pi)$, which is also illustrated in Figure 7.5 for the special case of the critical state model. Thus, the integral in Eq. (7.1) can be split into the integrals $0 \rightarrow \pi$ and $\pi \rightarrow 2\pi$ and rewritten as

$$T_{Hn} = \frac{1}{\pi}\int_0^{\pi} \phi(\theta) e^{-in\theta}\left[1 - e^{-in\pi}\right] d\theta,$$

so $T_{Hn} = 0$ for even n. Because the signal is also found to decrease rapidly with increasing harmonics, only the fundamental and third harmonic frequencies are studied.

7.2 The Sample

Figure 7.1 shows the geometry of the columnar defects in the $YBa_2Cu_3O_7$ single crystal used for this work. The columnar defects are created in the sample used in this

work by irradiating the crystal with 1.0GeV Pb ions. It has been shown[8] that these ions create columnar regions of amorphous material approximately 70Å diameter through the entire sample. The columnar shape of these defects makes them ideal pinning sites for flux lines parallel to the columns, and strongly enhanced pinning has been observed in samples with columnar defects as evidenced by enhanced critical currents.[8-10] However, the pinning strength has a very strong angular dependence as the magnetic field is rotated away from the direction of the columns. The angle θ defines the angle between the crystal c-axis and the applied dc magnetic field H as shown in Figure 7.1.

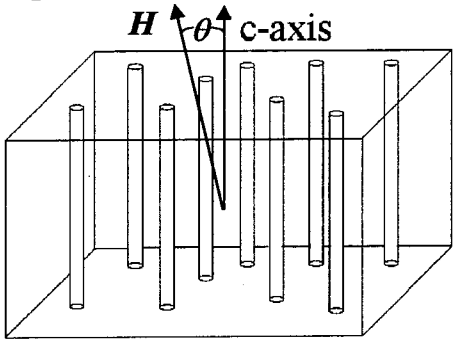


Figure 7.1. $YBa_2Cu_3O_7$ single crystal with columns of amorphous material created by heavy ion irradiation parallel to the crystal c-axis. The magnetic field H is applied at an angle θ relative to the crystal c-axis.

7.3 Experimental Setup

Although measurement of the flux ϕ is accomplished in susceptibility measurements by using a pickup coil (see Chapter 6), a more direct and more sensitive measurement is to place a miniature Hall probe directly under the sample to measure the local magnetic field. Additionally, the Hall probe sensitivity is independent of frequency so it is ideally suited for performing third harmonic measurements. For this Hall probe to work, its active region must be smaller than the sample to be measured (typical sample dimensions are 0.5mm × 0.5mm × 20μm), to ensure that it will not pick up excessive stray field passing around the sample. Such a Hall probe was developed and constructed by Dr. Marcin Konczykowski[20] at Ecole Polytechnique in Palaiseau, France. Before discussing how the Hall probe is used in performing the transmissivity measurements, it is important to consider the characteristics of the Hall probe itself and its response to the magnetic flux.

The most important characteristics for the Hall probe is small size, large Hall coefficient (R_H) for high sensitivity, and a temperature and field independent Hall coefficient. These requirements are all met by the Hall probe used in this work which is a 80μm × 80μm × 200μm piece of InSb lightly n-doped to a density $n \approx 10^{14}$ cm^{-3}. The Hall coefficient is given by $R_H = 1/(n_c e)$, where n_c is the carrier density, and e is the electron charge, so to maintain a temperature independent Hall coefficient the carrier density n_c must be temperature independent. In order to keep the carrier concentration n_c constant at low temperature, the doping level (n) must be high enough to ensure that the Bohr radii a_0 from each of the donor atoms overlap. If this condition is not met, the conduction electrons are isolated at each impurity site and will not form a continuous conduction band across the sample. In this low doping limit, the free carriers are only those electrons that are thermally activated into a conduction band and the carrier density will show the strong temperature dependence indicated in Figure 7.2. On the other hand, at a higher doping level, $n > \left(1/a_0\right)^3$, the Bohr radii will overlap and all donor electrons contribute to the conduction band giving a fixed carrier concentration $n_c = n$. At high temperatures, the thermal activation of electrons across the band gap will still lead to a large increase in carrier density with increasing temperatures, but this change in carrier density occurs only at temperatures T>~120K, well above the temperatures of interest in these experiments.

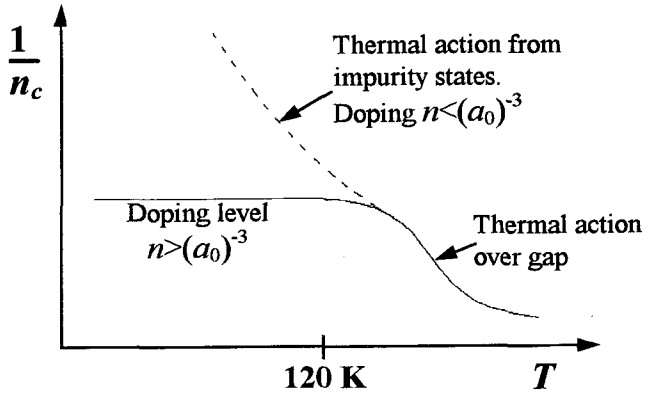


Figure 7.2. Inverse carrier density (n_c) versus temperature (T). The solid line shows the carrier density found in the Hall probes with a doping level $n>(a_0)^{-3}$ where a_0 is the Bohr radius. The dashed line shows the diverging carrier density for insufficient doping, $n<(a_0)^{-3}$.

Another problem that occurs at very low temperatures (<~ 20K) is that there are no phonon modes to scatter the electrons so the resistance drops and R_H becomes considerably smaller. However, over the temperature range of interest in these experiments, the Hall coefficient remains constant.

In order to maintain a large Hall coefficient R_H (which drops with increasing n_c), it is important not to dope the Hall probe too heavily. The optimized doping level is given by $n \approx (a_0)^{-3}$, the minimum level for which the Bohr radii will overlap. This doping level can be optimized by selecting a material which will give a very large effective Bohr radius for the doping atoms which is given by $a_0 \sim 1/m^*\varepsilon$ where m^* is the effective mass and ε is the dielectric constant. Because a small energy gap generally leads to a small effective mass,[21] InSb is chosen for its small band gap of only 0.23eV as compared to 1.52eV for GaAs, giving an effective mass m^* of $0.015m_e$ versus $0.066m_e$ for GaAs. Thus, InSb requires a doping level of only $n \approx 10^{14}$, while GaAs would require $n \approx 10^{16}$ making it 100 times less sensitive! Typical sensitivity for an InSb probe is a Hall coefficient $R_H \approx 70 m\Omega/Gauss$.

In addition to having R_H large and independent of temperature, it is also important to ensure that it is independent of the applied field B. Experiments are performed in dc magnetic fields up to ~2 Tesla, so in order to avoid complicated calibration procedures, it is important to have R_H be constant over this large field range. Furthermore, if R_H is field dependent, the Hall voltage $V_H \equiv I_{DC} R_H B$ (I_{DC} is the dc current injected into the Hall probe) would become nonlinear in the applied ac field, resulting in additional harmonic signals created by the Hall probe itself which would be nearly impossible to separate from the desired sample transmissivity signal T_{Hn}. Fortunately, maintaining a field independent R_H is easier than might be expected. In the InSb probes, it is found that at very low fields

(~100 Gauss), all of the conduction electrons are already in the lowest Landau level. This prevents the Schubnikow-de Haas effect in which the resistivity varies periodically with $1/H$ due to the evacuation of higher Landau levels with increasing magnetic field. Another concern would be that in the presence of large magnetic fields, the cyclotron radius $r_c \equiv v_F/\omega_c$, where v_F is the Fermi velocity and $\omega_c \equiv eH/m^*$ is the cyclotron frequency, would become smaller than the separation between doping atoms $\left(r_c < n^{-1/3}\right)$, effectively reducing the carrier concentration. Although the magnetic field at which this should occur is H~2 Tesla, no such effects are observed in this or higher fields.[22]

The only problem found in using the Hall probes in high magnetic fields is an observed saturation in the Hall voltage if it exceeds 100mV~300mV, thereby limiting sensitivity in high fields. To illustrate this problem, assume the desired sensitivity to a 1 Oe ac field is $10\mu\text{V} = \left(R_H \cdot 1\text{Oe} \cdot I_{DC}\right)$, then the dc current passed through the Hall probe must be $I_{DC} \geq 0.1\text{mA}$. However, with a 4 Tesla dc field applied, the voltage across the Hall probe will be $V_{Hall} = \left(R_H \cdot 0.1\text{mA} \cdot 4 \times 10^4\ \text{Oe}\right) = 280\,\text{mV}$, which risks saturating the Hall Voltage. In order to avoid saturation, I_{DC} must be reduced, which reduces the sensitivity to the ac field. The source of this saturation is attributed to the creation of "hot" electrons by the relatively large electric field. The electric field increases the kinetic energy of the free electrons sufficiently to change the mobility (because the relaxation time depends on the average electron energy), and these high energy electrons can also cause avalanche ionization in the valence band or impurity level. This effect can be described as an increase in the effective temperature of the electron gas, raising it higher than the actual temperature of the Hall probe. These hot carriers have a long mean free path, leading to a low resistance, thereby creating an effective short across the Hall probe which limits the maximum voltage it can generate.

A typical setup for the Hall probe magnetometer is shown in Figure 7.3. The electrical contacts to the Hall probe are 20μm diameter gold wires bonded to the InSb by discharging a 1000μF capacitor charged to 9V through the wires and Hall probe. The electric discharge spot welds the wire in place. With the wires attached, the Hall probe is anchored to a sapphire substrate with GE varnish and covered with a small piece of mica to protect the leads. A small copper coil is attached to the sapphire around the Hall probe as shown in Figure 7.3. This coil is similar to those used for susceptibility measurements as described in Chapter 6.2, with an inner diameter of 0.125 inches made from 24 turns of 0.002 inch diameter copper wire with an inductance of ~10μH. Although a larger coil would provide a more uniform ac field, the small size is chosen to minimize both the coil inductance and the inductive coupling between the coil and Hall probe, M_{13}. (The notation M_{13} is used to remain consistent with the notation of the one-loop model discussed in section 7.4.) This inductive coupling is the primary limitation to performing measurements at higher frequencies because it introduces a background voltage given by $V_{back} = 2\pi f M_{13} I_1$ which is independent of the sample. The coupling M_{13} can also be reduced by positioning the Hall probe leads in such a way as to minimize the area

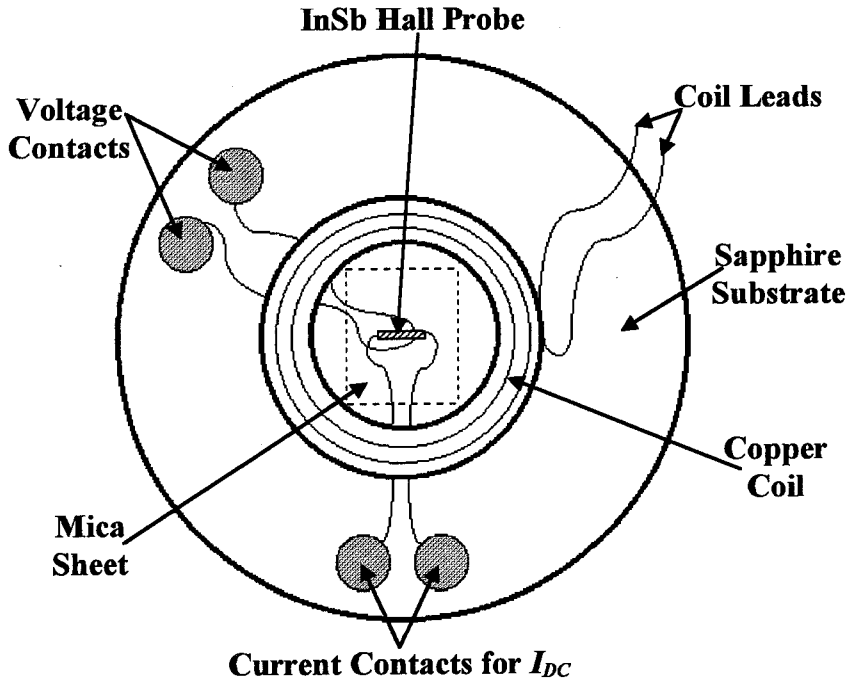


Figure 7.3. The Hall probe magnetometer. The Hall probe is shown in the center of the coil which provides the ac excitation field. A piece of mica is placed over the Hall probe to protect the wires when samples are loaded and unloaded. The sample is on the mica over the Hall probe.

of the loop created by the voltage path, but this background is still found to dominate the signal for frequencies f>~10kHz.

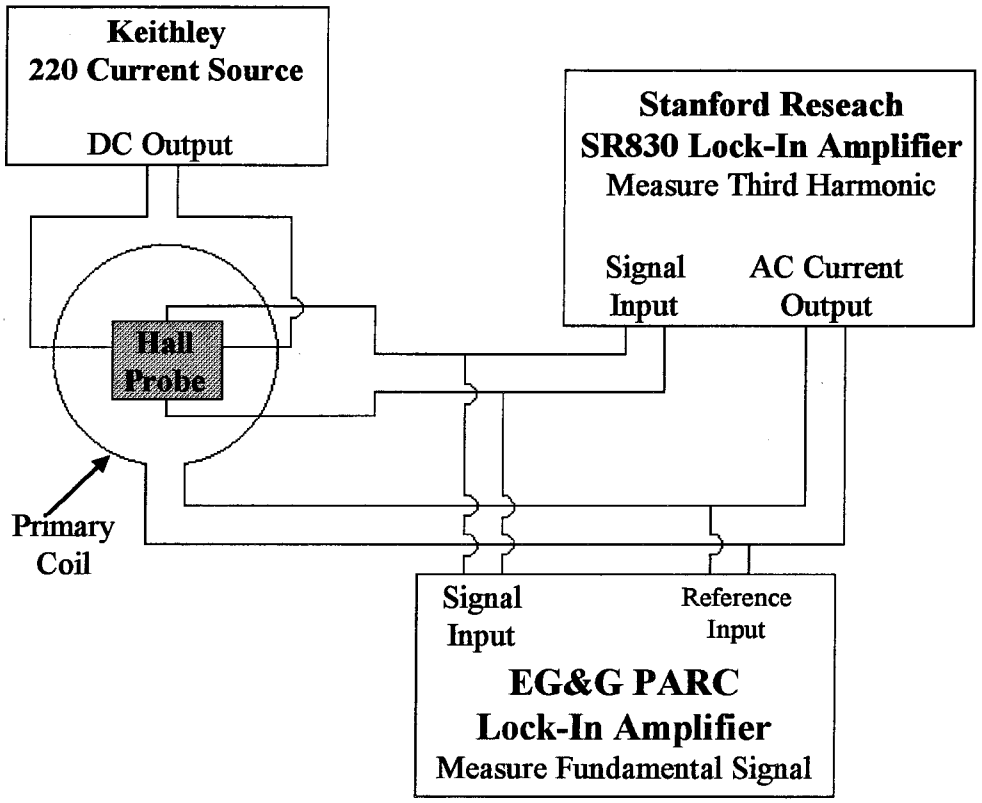


Figure 7.4. Block diagram showing the electronic instrumentation used for measuring the Hall probe signal.

The Hall probe signal is measured using two lock-in amplifiers as shown in Figure 7.4. A dc current is passed through the Hall probe using a Keithley 220 current source, and the primary coil is driven with an ac current from the output of the SR830 lock-in amplifier. The ac voltage from the Hall probe is measured by two different lock-in amplifiers. The SR830 lock-in amplifier is able to directly measure the third harmonic component relative to the ac output signal. The EG&G lock-in amplifier measures the Hall probe signal at the fundamental frequency as referenced to the ac output driving the primary coil. Although this measurement could be performed with a single lock-in amplifier that first measures the fundamental signal and then the third harmonic signal, there is a considerable

delay in waiting for the signal to stabilize for each of the two measurements. By measuring the two signals simultaneously with separate lock-in amplifiers, the stabilization time is minimized and measurements can be taken much more quickly.

7.4 One-Loop Model for Transmissivity

A model for the transmissivity has been made based on approximating the sample response as a single current loop.[15] In this model, the system is divided into three components: the primary coil, the sample, and the detector. The primary coil is driven by a current $I_1(t) = I_m \cos(\omega t)$ where $\omega = 2\pi f$ is the angular frequency. It is assumed that the components are linked by a simple mutual inductance so that the flux in the sample (Φ_2) and in the detector (Φ_3) can be given by:

$$\begin{aligned} \Phi_2 &= M_{12} I_1 + L_2 I_2 \\ \Phi_3 &= M_{13} I_1 + M_{23} I_2 \end{aligned} \tag{7.2}$$

where M_{12} is the mutual inductance between the primary coil and the sample, L_2 is the self inductance of the sample, I_2 is the current in the sample, M_{23} is the mutual inductance between the primary coil and the detector, and M_{23} is the mutual inductance between the sample and the detector. The mutual inductance M_{13} is the coupling directly from the primary coil to the detector, independent of the sample, and acts only to create a background signal. The Hall probe detector has a moderately high resistance and therefore forms no current loop to modify the flux Φ_2 or Φ_3. It is convenient to define a normalized flux ϕ penetrating through the sample:[15]

$$\phi \equiv \left(M_{12} I_m\right)^{-1} \Phi_2 = \cos(\omega t) + \frac{L_2}{M_{12} I_m} I_2 . \tag{7.3}$$

We also define a normalized sample current:

$$j_2 = \frac{L_2}{M_{12} I_m} I_2 , \tag{7.4}$$

so that $\phi = \cos(\omega t) + j_2$. In order to use Eqs. (7.1)-(7.3) to develop an expression for the transmissivity T_H, additional assumptions must be made on the form of M_{12} and L_2. One approach which has been used recently is to apply a critical state model[14,15] to estimate these values. A more accurate description is developed which incorporates a relaxation time τ for the flux lines, and a crossover current j which is similar to an effective depinning current below which the vortices are fixed, and above which vortices move with a relaxation rate τ^{-1}.

7.5 Critical State Model

In the critical state model, the current I_2 is limited to a maximum critical current I_c. For currents $I_2<I_c$, the sample will maintain constant flux Φ_2 which corresponds to zero dissipation $E \propto \frac{d\Phi_2}{dt} = 0$ as shown in Figure 7.5. We also define a normalized critical current density $j_c = \left(L_2/\left(M_{12}I_m\right)\right)I_c$. Using this model, analytic expressions for both the fundamental and higher harmonic transmissivity can be obtained:[15]

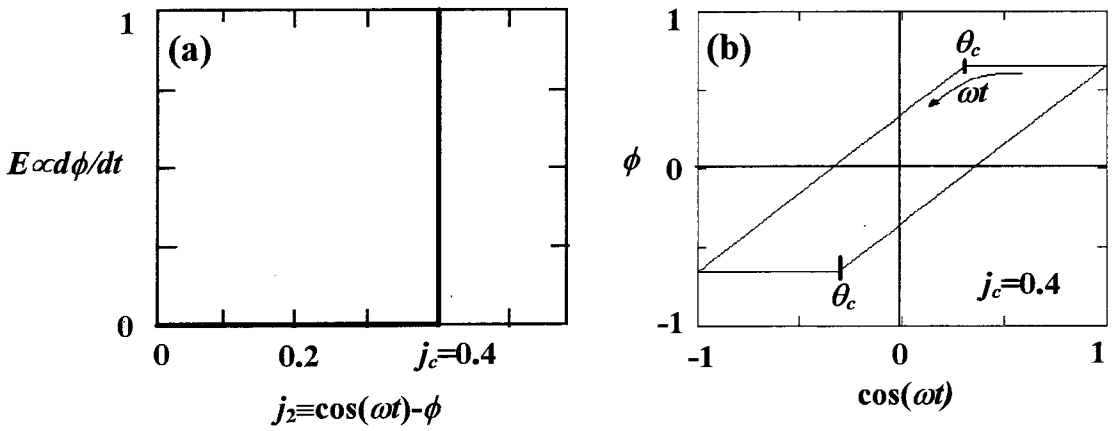


Figure 7.5. Critical state model with $j_c = 0.4$, showing **(a)** voltage vs. current characteristic requiring $j_2<j_c$, and **(b)** sample flux ϕ as a function of time through one complete cycle $\omega t = 0 \rightarrow 2\pi$. The direction of increasing time is in the counter-clockwise direction as indicated by the arrow.

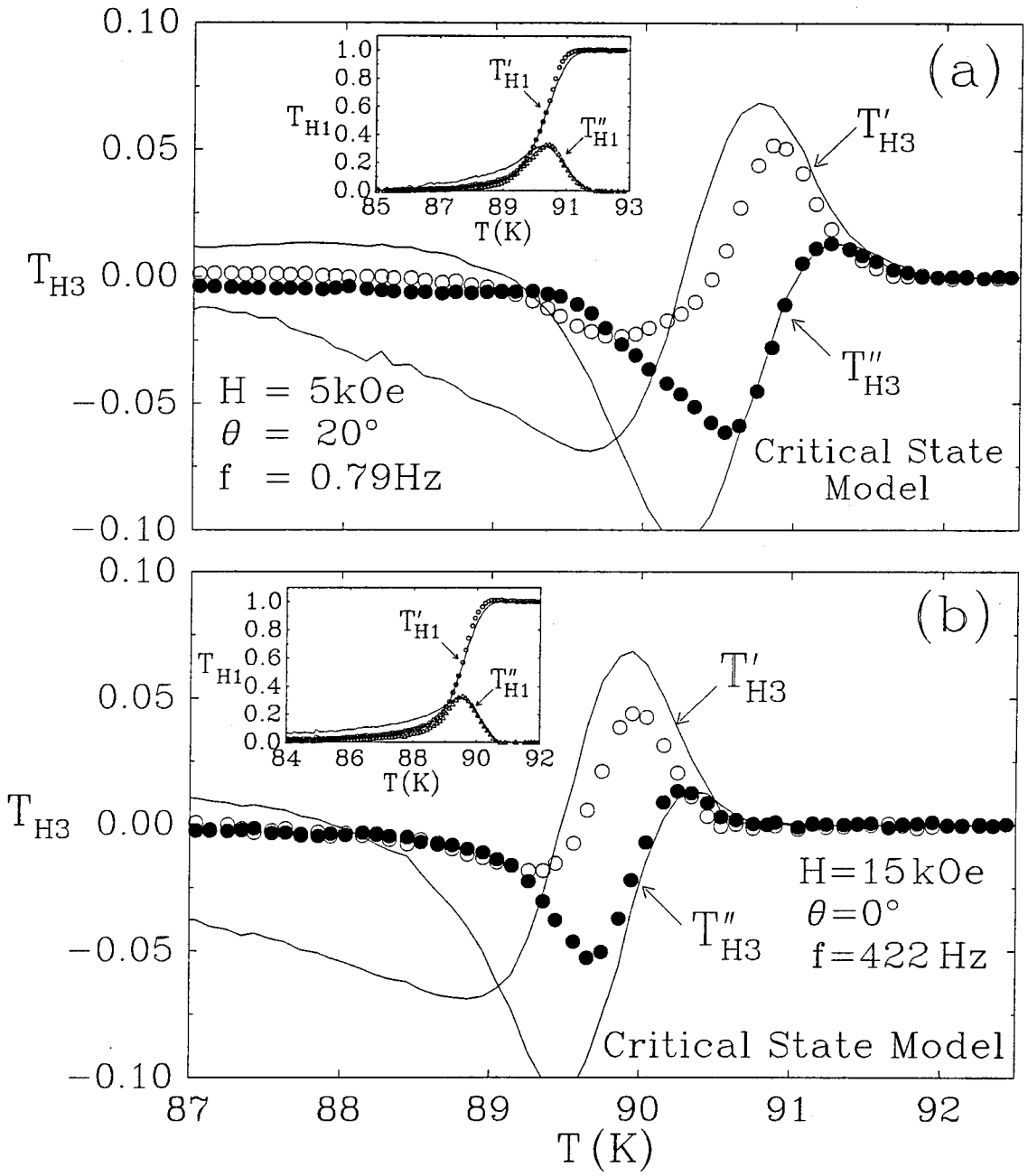


Figure 7.6. Real (T'_{H3}) and imaginary (T''_{H3}) parts of the third harmonic transmissivity versus temperature (T) data for **(a)** $\theta = 0°$, $f = 7.7$Hz, $H = 6$kG, and **(b)** $\theta = 90°$, $f = 7.7$Hz, $H = 15$kG. The solid lines are theoretical fits using the critical state model, Eq. (7.5)-(7.6). The insets show the corresponding fundamental transmissivity $(T_{H1} = T'_{H1} - iT''_{H1})$ vs. T data with the theoretical fits.

$$T_{H1} \equiv T'_{H1} - iT''_{H1} = 1 - \frac{\theta_c}{\pi} - \frac{i}{2\pi}\left(1 - e^{-1i\theta_c}\right) \qquad \text{(for n = 1)} \tag{7.5}$$

$$T_{Hn} \equiv T'_{Hn} - iT''_{Hn} = \frac{2i}{n\left(n^2-1\right)\pi}\left[1 - \left(\cos\theta_c + in\sin\theta_c\right)e^{-in\theta_c}\right] \qquad \text{(for odd n>1)} \tag{7.6}$$

where $\cos\theta_c \equiv 1 - 2j_c$ is the phase at which the current crosses from the subcritical state ($j_2 < j_c$) to the critical state $(j_2 = j_c)$ as indicated in Figure 7.5. By fitting the real part of the fundamental transmissivity data $T'_{H1}(T)$ to Eq. (7.5), and solving for $j_c(T)$, the inset of Figure 7.6 shows that excellent agreement can be found for both the real and imaginary parts of the fundamental transmissivity T_{H1}. However, inserting this $j_c(T)$ into Eq. (7.6) shows agreement with the third harmonic data only at temperatures very close to the onset of T_{H3}, as shown in Figure 7.6. Furthermore, the critical state model contains no frequency dependence or orientational dependence for anisotropic superconductors, so it fails to provide a consistent description of the experimental data as shown in Figure 7.7. This discrepancy between the critical state model and experimental data can be seen only in measurements of the higher harmonics of the transmissivity, which emphasizes the importance of performing these third harmonic measurements for better understanding of nonlinear vortex dynamics.

7.6 Relaxational "τ-j" Model

In order to better understand the third harmonic transmissivity, we use an alternate model which incorporates a relaxation time τ in the vortex motion. This time scale is roughly defined through the relation $\frac{d}{dt}\phi \sim \frac{1}{\tau}\left(\cos(\omega t) - \phi\right)$, where $\cos(\omega t)$ is the normalized flux in the absence of a sample. However, we expect some pinning barrier so that for sample currents $j_2 \equiv \cos(\omega t) - \phi$ below a crossover current j, the flux would remain constant and only currents above that crossover current $j_2 > j$ would contribute to flux motion.

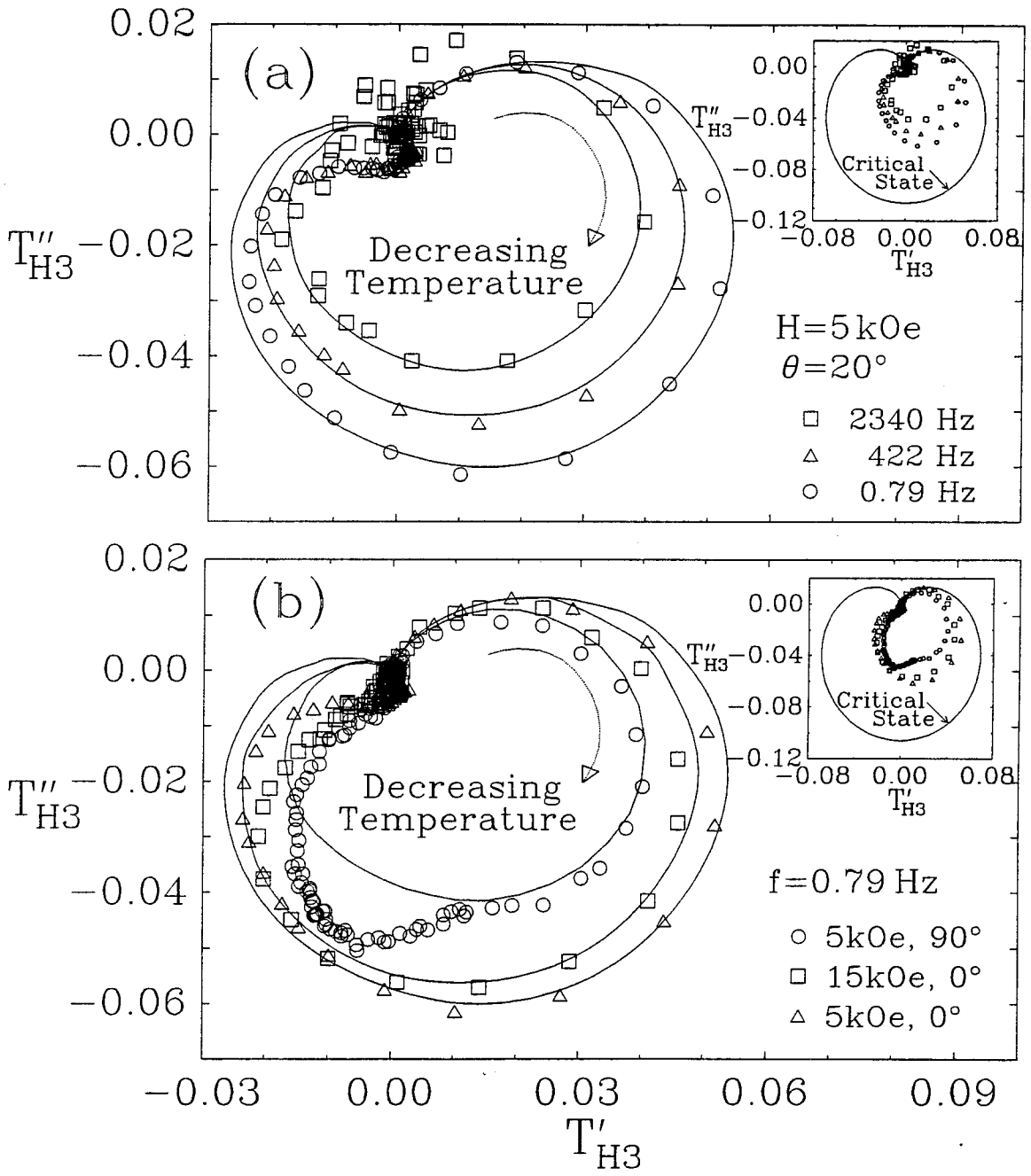


Figure 7.7. The complex plane third harmonic data (T''_{H3} vs. T'_{H3}) for **(a)** various frequencies (f) and **(b)** various angles (θ). The solid lines are the theoretical fits using the "τ-j" model. Note that the origin indicates the onset of T_{H3}; with decreasing temperature the data follows the clockwise direction. The insets show a comparison to the critical state model.

Including the crossover current j leads to equations for the vortex motion:

$$\begin{aligned} \frac{d}{dt}\phi &= \frac{1}{\tau}\left(\cos(\omega t) - \phi \pm j\right) && \text{for } |\cos(\omega t)-\phi|>j \\ \frac{d}{dt}\phi &= 0 && \text{for } |\cos(\omega t)-\phi|<j \end{aligned} \tag{7.7}$$

By equating $j_2 \equiv \cos(\omega t) - \phi$ and $R_2 \equiv \tau^{-1}$, this "τ-j" model is mathematically equivalent to the "flux flow" model used in reference 15. The "flux flow" model and Eq. (7.7) both predict a specific current-voltage relation as indicated in Figure 7.8(a), but fitting to the model is not meant to suggest that the true current-voltage relation follows this simple form. Instead, as indicated in Figure 7.8(a), this is only a linear approximation which best interprets the sample characteristics in terms of a single relaxation time τ. We note that j and τ are expected to increase with increasing amplitude of the applied ac field because the higher current portion of sample signal shown in Figure 7.8 will be measured. The fitting would therefore have to be adjusted to fit the higher curvature of that section of the curve. This amplitude dependence is consistent with experimental observations.

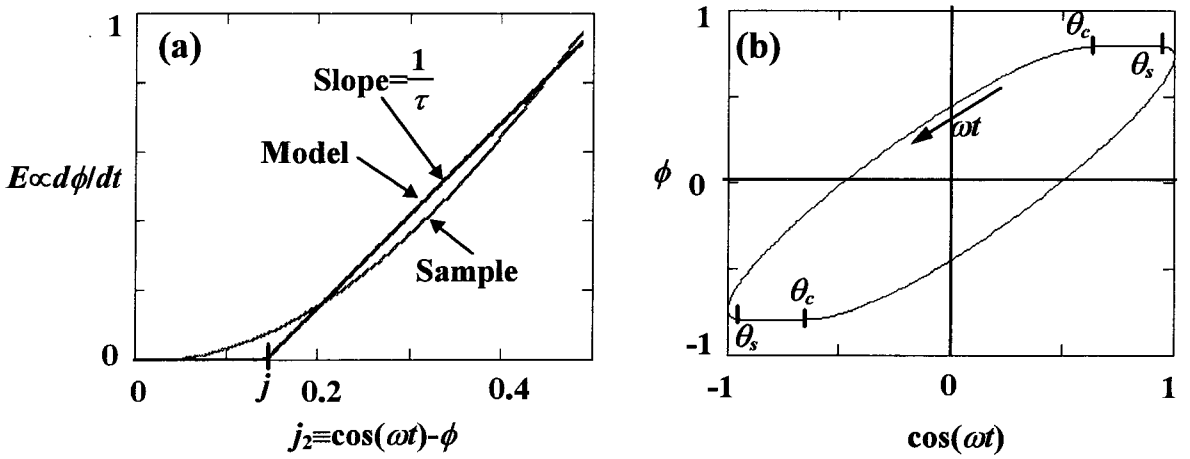


Figure 7.8. **(a)** Voltage vs. current plot showing a simulated sample curve and a fit using the single relaxation time $\tau = 0.37$, and current $j = 0.15$. **(b)** The flux (ϕ) as a function of time through one complete cycle $\omega t = 0 \rightarrow 2\pi$ using the same parameters.

We define a characteristic phase θ_c to be the phase at which the sample current equals the crossover current $j_2 = j$ and the phase θ_s represents a phase lag due to the presence of finite relaxation time τ. We note that for the limit $\theta_s \to 0$, or equivalently $\tau = 0$, this τ-j model reduces to the critical state model. The values of θ_s and θ_c can be more easily understood from Figure 7.8(b) and Figure 7.9. The phase angles θ_s and θ_c are directly related to τ and j through the following relations:[15]

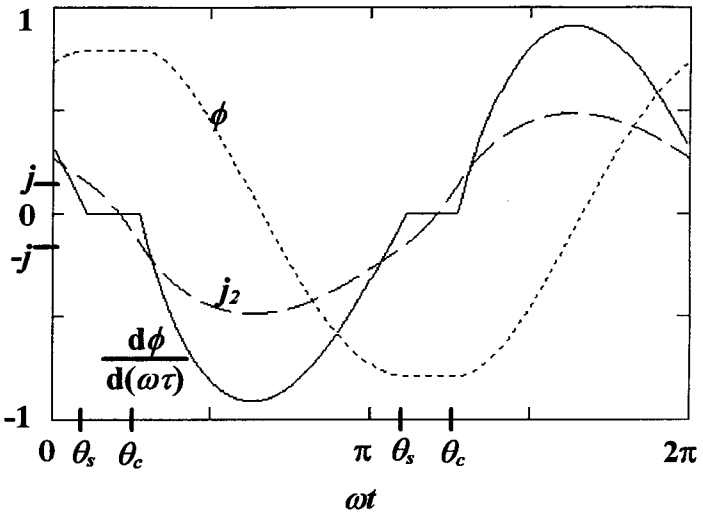


Figure 7.9. The time dependence of the normalized flux ϕ, sample current j_2, and electric field $E = d\phi/dt$, shown through one complete cycle $\omega t = 0 \to 2\pi$.

$$\cos\theta_c = \cos\theta_s - 2j$$
$$\frac{\omega\tau\cos\theta_c - \sin\theta_c}{\omega\tau\cos\theta_s - \sin\theta_c} = -e^{(\pi+\theta_c-\theta_s)/(\omega\tau)}, \tag{7.8}$$

so we can numerically solve for $\tau(\theta_s,\theta_c)$ and $j(\theta_s,\theta_c)$. Analytic expressions can also be found for the transmissivity:[15]

$$T_{H1} \equiv T'_{H1} - iT''_{H1} = \frac{1}{(1+i\omega\tau)}\left[1 - \frac{1}{\pi}(\theta_c - \theta_s) - \frac{i}{2\pi}\left(e^{-2i\theta_s} - e^{-2i\theta_c}\right)\right]$$
$$T_{H3} \equiv T'_{H3} - iT''_{H3} = \frac{i}{12\pi(1+i3\omega\tau)}\left[(\cos\theta_s + i3\sin\theta_s)e^{-i3\theta_s} - (\cos\theta_c + i3\sin\theta_c)e^{-i3\theta_c}\right] \tag{7.9}$$

Using the expression for $\tau(\theta_s,\theta_c)$, the transmissivity data $T_{H1}(T)$ and $T_{H3}(T)$ is fit to Eq. (7.9) to obtain values of $\theta_c(T)$ and $\theta_s(T)$. From the fitting in Figure 7.10 we see excellent agreement for the third harmonic transmissivity and reasonably good agreement for the fundamental transmissivity. Additionally, the fitting in Figure 7.7 shows that data taken at various frequencies and magnetic field orientations θ can all be well described by the

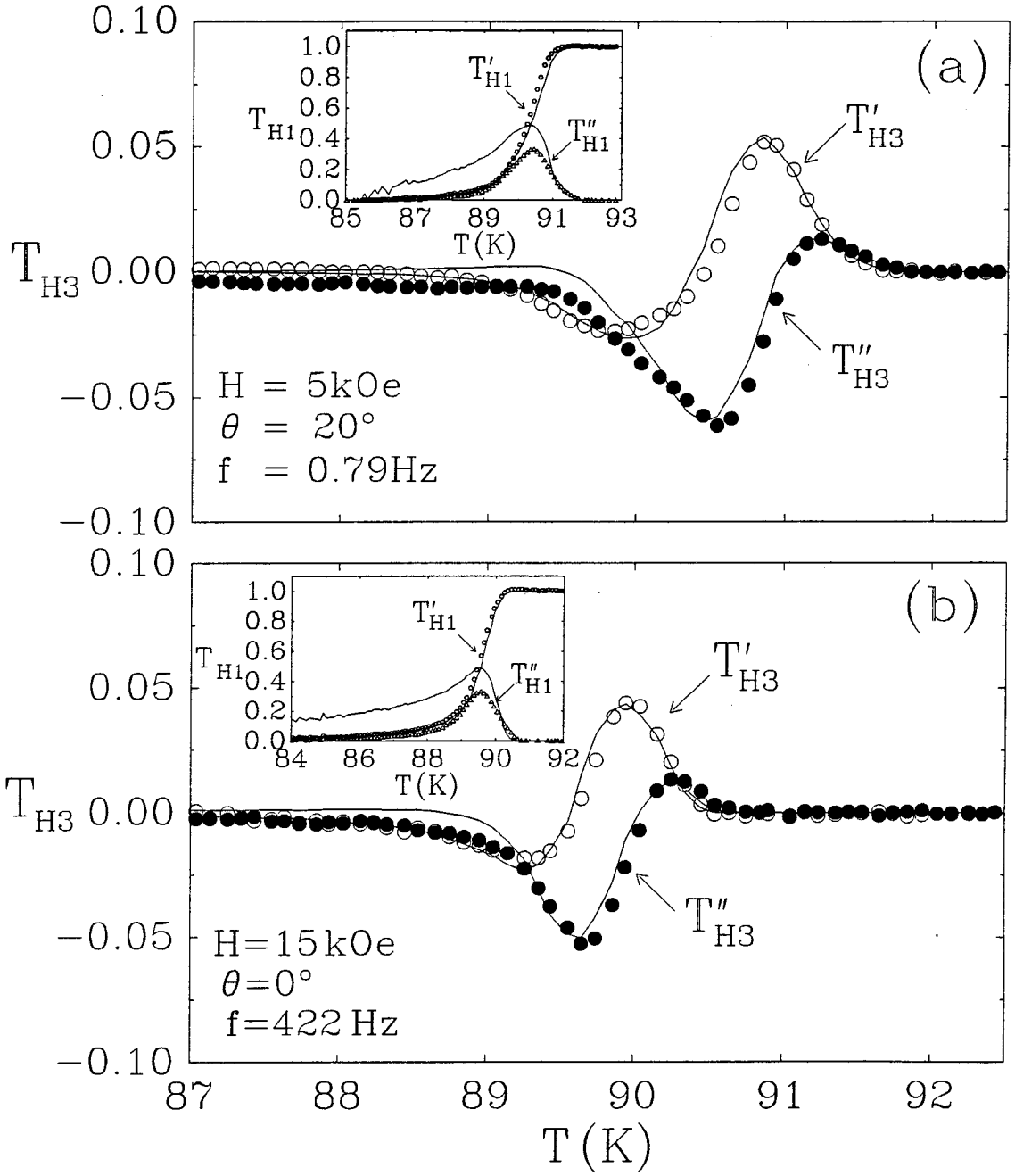


Figure 7.10. Comparison of the same T_{H3} vs. T data shown in Figure 7.6 using the τ-j model (solid lines) for **(a)** $H = 5$kOe, $\theta = 20°$, and $f = 0.79$Hz, and **(b)** $H = 15$kOe, $\theta = 0°$, and $f = 422$Hz. The corresponding fundamental transmissivity (T_{H3}) is shown in the insets. The third harmonic (T_{H3}) data shows excellent agreement with this model at all temperatures in contrast to the critical state model.

parameters τ and j in Eq. (7.9). Here θ denotes the angle between the applied magnetic field and the sample c-axis as shown in Figure 7.1.

7.7 Experimental Results

By incorporating both a finite relaxation time τ and crossover current j, we find that the data taken at various frequencies f and magnetic field orientations θ can be well described with the use of Eq. (7.9). The solid lines in the complex plane T''_{H3} vs. T'_{H3} data in Figure 7.7 are the theoretical curves derived from the numerical analysis using Eq. (7.9) and show excellent agreement with data except for slight deviations at $\theta = 90°$ and very low temperatures. In contrast, the critical state model which assumes $\tau = 0$ and a single fitting parameter (j_c)[14,15] does not provide proper description for the third harmonic data except for a narrow temperature window near the onset of T_{H3}, as shown in Figure 7.6. Furthermore, the critical state model does not allow any frequency dependence of field orientational dependence as is shown in the insets of Figure 7.7. The results in Figure 7.6, Figure 7.7, and Figure 7.10 underscore the importance of considering a temperature dependent characteristic relaxation time τ. The τ versus T data at various frequencies (f) and angles (θ) are shown in Figure 7.12(a) and (b).

Next, we consider the physical significance of the crossover current j. The E vs. j_2 curve as depicted in Figure 7.8(a) represents the sample response to a given set of experimental conditions (a given T, H, θ, and f). For temperatures near the Bose-glass transition, the linear approximation in Figure 7.8(a) can provide a reasonable estimate for a characteristic sample resistivity ρ and an effective "depinning" current density j, with $j \to 0$ for $T >> T_{BG}$. However, this approximation no longer provides a good description for the E vs. j_2 isotherms if $T << T_{BG}$ because of the exponentially vanishing E with increasing J. In this low temperature limit, the fitting parameter j can no longer be interpreted as an effective depinning current. As shown in Figure 7.11, for temperatures just below the onset temperature of T_{H3}, the fitting parameter j first increases with decreasing T, reaches a

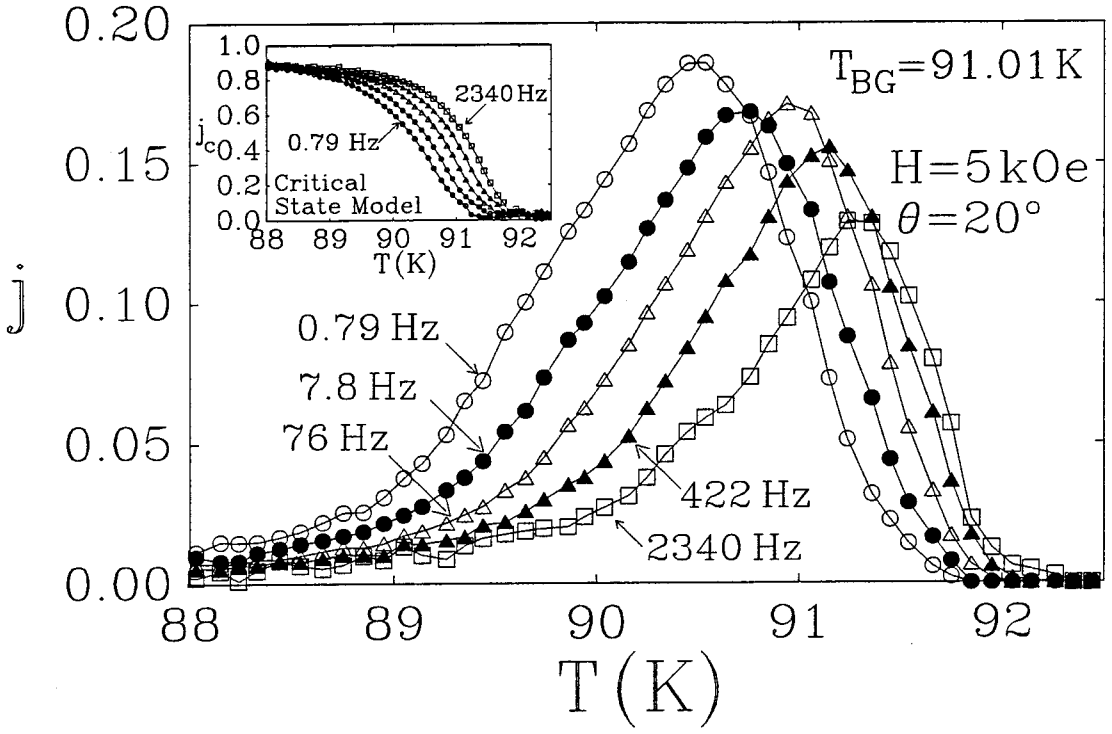


Figure 7.11. The normalized induced current density j vs. T data for various frequencies. The inset shows the j vs. T curves derived from the critical state model.[14,15]

maximum, and then decreases again at low temperatures. It is worth noting that the high temperature part of the $j(T,f)$ vs. T curves follows the frequency dependence for the vortex-liquid state:[7]

$$j(\omega) = j_c\left[1-\left(\frac{k_B T}{U_{pl}}\right)\ln\left(\frac{1}{2\pi f\tau}\right)\right] \tag{7.10}$$

with j_c being the normalized critical current density, and U_{pl} is the effective activation barrier in the liquid state. However, for temperatures well into the Bose-glass state, the rapid decease of $j(f,T)$ with decreasing T in Figure 7.11 is largely due to the limitation of our "τ-j" approximation, and is therefore not realistic. A better approximation for understanding the low temperature third harmonic transmissivity is the flux creep model given in reference 15, which assumes $E \propto J^{\sigma}$ with the single fitting parameter σ. Such an ap-

proximation has recently been employed to yield information about vortices in the Bose-glass state.[14]

Another precaution that should be taken in using this τ-j analysis is that the amplitude of the ac field, h_0, should be restricted to the range where the parameter τ does not change significantly with h_0. The parameter j will naturally increase with h_0 due to the increasing induced current I_2 in the sample. In this work, measurements have been performed at two different h_0 values h_0=1.0Oe and h_0=0.4Oe, and the corresponding results for τ (to be described below) are found to be consistent for both amplitudes.

7.8 Critical Scaling Analysis

Once we justify the use of both τ and j to fully account for the quantitative behavior of T_{H3}, the next task is to relate the derived information to Bose-glass theory.[3] To use the results of the third harmonic transmissivity to study the Bose-glass to superfluid phase transition which has been found in samples with columnar defects, we must consider the scaling laws discussed in Chapter 3. From the fitting of T_{H3}, we obtain a vortex relaxation rate $\tau^{-1} \sim \mathrm{d}\phi/\mathrm{d}t$ versus frequency. This τ is **not** the critical relaxation time scale $\tau_T \sim \xi_\perp{}^{z'}$ with z' the dynamic exponent which describes the relaxation of dislocation loops. Instead, $\tau^{-1} \sim \mathrm{d}\phi/\mathrm{d}t$ describes the diffusion of the vortices into the sample and is related to the vortex diffusion time given by $\tau \sim \left(\ell^2/\rho\right) \sim \rho^{-1}$ where ρ is the sample resistivity and ℓ is the characteristic sample dimension. For the Bose-glass system the resistivity scaling relation is given by[6] $\rho \sim \xi_\perp^{-(z'-\zeta)}$ so that the scaling relations for τ are given by:

$$\tilde{\tau}(\tilde{f}) \equiv \tau \left|1-\frac{T}{T_{BG}}\right|^{\nu_\perp(z'-\zeta)}, \qquad \tilde{f} \equiv \left|1-\frac{T}{T_{BG}}\right|^{-\nu_\perp z'} \tag{7.11}$$

where T_{BG} is the Bose-glass transition temperature, $\nu_\perp>0$ and $\nu_\parallel>0$ are the transverse and longitudinal static exponents, and the ratio of the exponents is given by $\zeta \equiv \nu_\parallel/\nu_\perp$. In

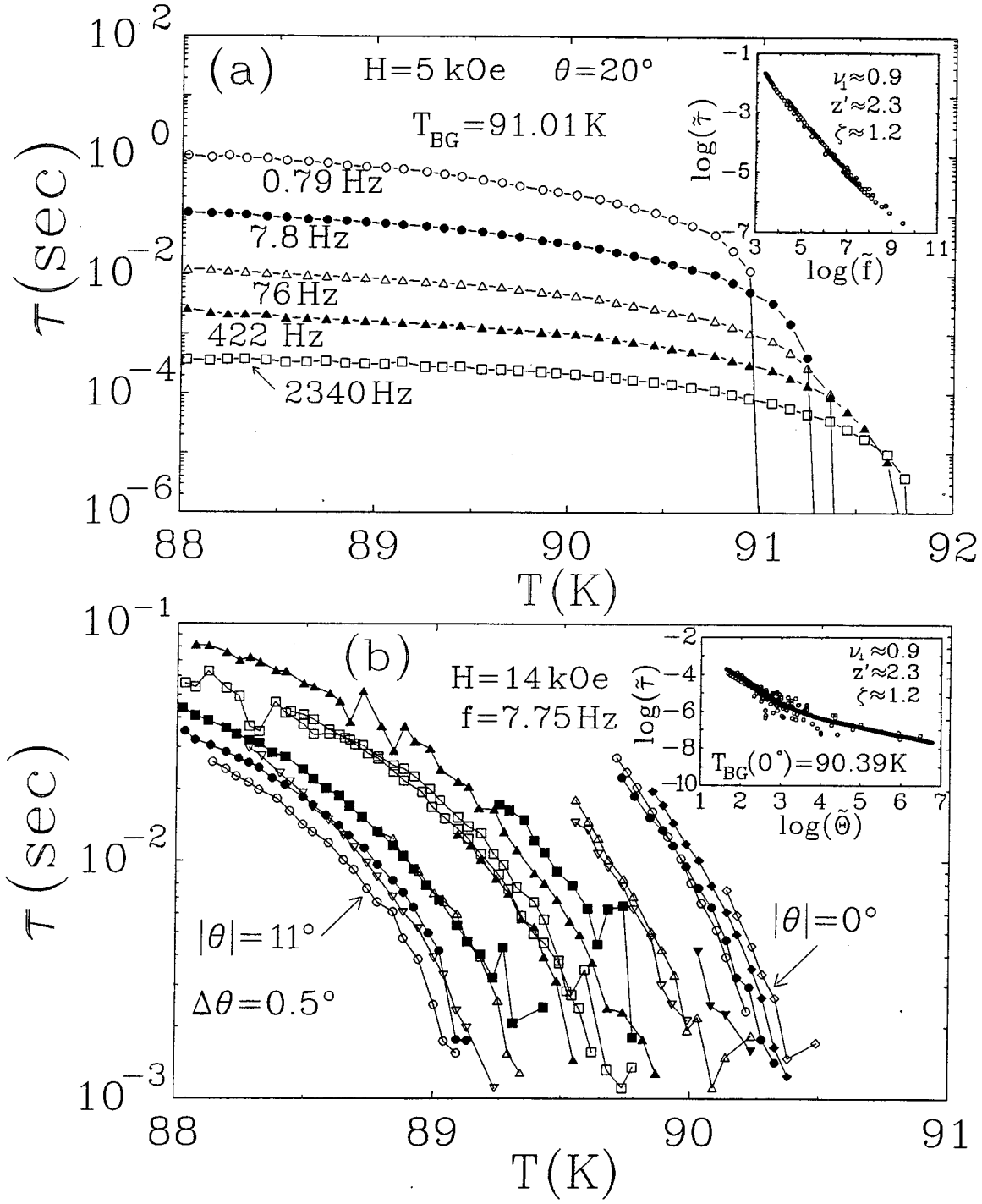


Figure 7.12. **(a)** The characteristic relaxation τ vs. T data for various frequencies. The inset shows the scaling function $\tilde{\tau}(\tilde{f})$ using Eq. (7.11). **(b)** τ vs. T data for various angles and f = 7.9 Hz. The inset shows the scaling function $\tilde{\tau}(\tilde{\Theta})$ obtained from the data for $T<T_{BG}$ using Eq. (7.12).

addition, the critical relaxation rate (τ_T^{-1}) of vortex fluctuations near T_{BG} is given by $\tau_T^{-1} \propto \xi_\perp^{-z'}$, where z' is the dynamic exponent.

We note that Eq. (7.11) is applicable to all τ vs. T curves taken at any θ smaller than the accommodation angle[5,6] ($\theta_{cr} \approx 30°$) of the columnar defects. Figure 7.12(a) shows a series of τ versus T data taken at several different frequencies. By using Eq. (7.11), the exponents $\nu_\perp(z'-\zeta) = 0.9 \sim 1.1$ and $\nu_\perp z' = 2.0 \sim 2.3$, and transition temperatures $T_{BG}(\theta<\theta_{cr})$ can be found which optimized the "collapsing" of the τ vs. T data into the scaling function $\tilde{\tau}(\tilde{f})$ shown in the inset of Figure 7.12(a).

To derive all three critical exponents $\nu_\perp$, ζ and z', another scaling relation is needed. This scaling relation may be obtained by considering the angular dependence of τ as shown in Figure 7.12(b). Since the universality of the Bose-glass transition requires that all critical exponents be universal for $(\theta < \theta_{cr})$, and since θ is related to the effective transverse magnetic field component[3,5,6] $\Theta \equiv \sin\tilde{\theta} \sim \xi_\perp^{-1}\xi_\parallel^{-1} \sim \left|1-\left[T/T_{BG}(0°)\right]\right|^{\nu_\perp(\zeta+1)}$, we find that in the $f \to 0$ limit,

$$\tilde{\tau}(\tilde{\Theta}) \equiv \tau\left|1-\frac{T}{T_{BG}(0°)}\right|^{\nu_\perp(z'-\zeta)}, \quad \tilde{\Theta} \equiv \Theta\left|1-\frac{T}{T_{BG}(0°)}\right|^{-\nu_\perp(1+\zeta)}. \tag{7.12}$$

We note that for an applied dc field at an angle θ relative to the c-axis of an anisotropic superconductor, the effective transverse magnetic field component $\Theta \equiv \sin\tilde{\theta}$ is given by[23]

$$\sin\tilde{\theta} \equiv \frac{\varepsilon \sin\theta}{\sqrt{\cos^2\theta + \varepsilon^2 \sin^2\theta}} \tag{7.13}$$

where the anisotropic effective mass ratio ε^{-2} in the $YBa_2Cu_3O_7$ system is approximately $\varepsilon^{-2} \approx 60$.[24] Thus, by inserting the $T_{BG}(0°)$ value obtained from the analysis using Eq. (7.11) into Eq. (7.12), the angular dependent τ vs. T curves can be scaled into universal functions $\tilde{\tau}(\tilde{\theta})$, as exemplified in the inset of Figure 7.12(b). In addition to reconfirming

the product $\nu_\perp(z'-\zeta)$, we also obtain a second scaling relation for the exponents, $\nu_\perp(\zeta+1)\approx 2.2$. Combining the results derived from Eqs. (7.11)-(7.13), we have $\nu_\perp = 0.9\pm 0.2$, $\zeta = 1.2\pm 0.2$, and $z' = 2.3\pm 0.3$, in good agreement with the exponents derived from ac resistivity studies on a similar sample.[6] The success of the above critical scaling analysis underlines the importance of incorporating the characteristic relaxation time τ into the one-loop model, so that excellent agreement with the Bose-glass theory can be achieved by the studies of the third harmonic transmissivity.

In considering the angular dependence of the transition temperatures $T_{BG}(\theta)$, we can also rescale the transition temperatures into the isotropic frame by considering the relation:

$$H_{BG}(T,\theta) = H_{BG0}(\theta)\left|1-\frac{T}{T_{c0}}\right|^{2\nu_0} \tag{7.14}$$

where T_{c0} is the zero field transition temperature and ν_0 is the zero field critical exponent. The temperature in Eq. (7.14) can be rescaled into the isotropic frame $T\rightarrow\widetilde{T}$ by rescaling the coefficient $H_{BG0}(\theta)\rightarrow\widetilde{H}_{BG0}(\theta)$ at a fixed field H_{BG}(T), giving the relation

$$\widetilde{T} = T_{c0}+T_{c0}\left(\frac{H_{BG}}{\widetilde{H}_{BG0}}\right)^{1/2\nu_0} = T_{c0}+\frac{T-T_{c0}}{\left(\sqrt{\cos^2(\theta)+\varepsilon^2\sin^2(\theta)}\right)^{1/(2\nu_0)}}. \tag{7.15}$$

We find the value $\nu_0 = 0.5\pm 0.05$ by fitting Eq. (7.14) to the H_{BG} vs. T phase boundary shown in inset of Figure 7.13. After rescaling the temperature and field according to Eqs. (7.13) and (7.15), the resulting $\widetilde{T}$ vs. $\widetilde{\theta}$ phase diagram is depicted in Figure 7.13 for different magnetic fields. We note that for the very small angles $|\widetilde{\theta}|<2°$ shown in Figure 7.13, there is little change in the temperature when converting to the isotropic frame, so $\widetilde{T}\approx T$.

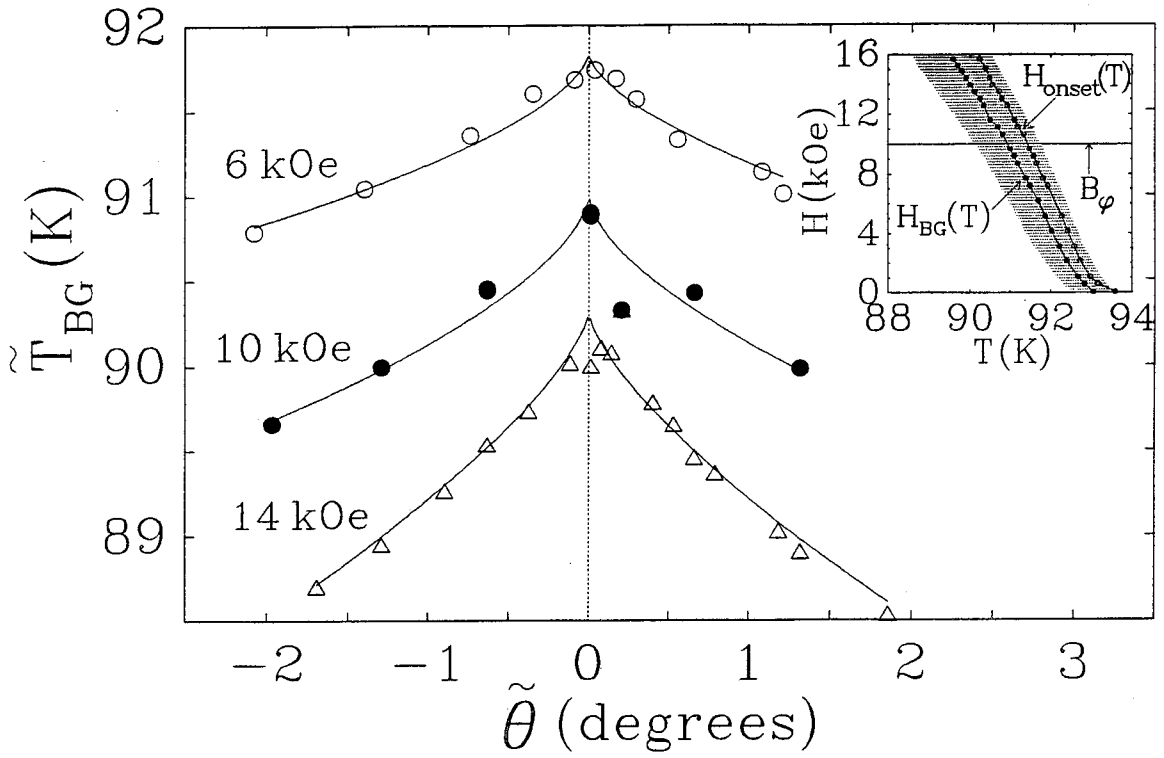


Figure 7.13. The angular dependent Bose-glass transition temperatures $\tilde{T}_{BG}(\tilde{\theta})$ is shown to follow the scaling law given in Eq. (7.16). The inset shows the Bose-glass phase diagram for $\theta = 0°$, determined from critical scaling of the T_{H3} data. The shaded region indicates the critical regime of $H_{BG}(T)$, and the line $H_{onset}(T)$ depicts the onset of T_{H3}.

To verify whether the derivation of $T_{BG}(\theta)$ from Eq. (7.12) is reasonable, we consider the following scaling relation for the angular dependent Bose-glass transition temperatures:[3,6]

$$\left[\tilde{T}_{BG}(0°) - \tilde{T}_{BG}(\tilde{\theta})\right] \sim \left(\sin\tilde{\theta}\right)^{1/[\nu_\perp(\zeta+1)]} \tag{7.16}$$

In Figure 7.13 we show the $\tilde{T}_{BG}(\tilde{\theta})$ vs. $\tilde{\theta}$ curves taken at three different dc magnetic fields, 6, 10 and 14 kOe. We find that the best fit with the use of Eq. (7.16) yields $\left[\tilde{T}_{BG}(0°) - \tilde{T}_{BG}(\tilde{\theta})\right] \sim \left(\sin\tilde{\theta}\right)^{0.5}$, which is in excellent agreement with the results $\zeta \approx 1.1$ and $\nu \approx 1.0$, once again confirming the consistency of our third harmonic data with the Bose-glass theory.[3,6]

It is worth comparing the above results with the fundamental transmissivity data T_{H1}. Since $\chi = (T_{H1} - 1)$, the same analysis used in Chapter 6 for the susceptibility can be used. In the linear response limit, χ follows the scaling relation[17] $\chi \sim \lambda^{-1}$, where[5] $\lambda \sim \rho_s^{-1/2} \sim \xi_{\|}^{1/2} \sim \delta^{-\nu_\perp \zeta/2}$ for the Bose-glass transition and ρ_s denotes the superfluid density.[2,5] Thus, the scaling relation for $\tilde{\chi}(\tilde{f})$ becomes

$$|\tilde{\chi}| = |\chi|\delta^{-\nu_\perp \zeta/2}, \qquad \tilde{f} = f\delta^{-\nu_\perp z'} \tag{7.17}$$

By applying Eq. (7.17) to the χ vs. T data taken at different frequencies, we obtain another scaling relation for the exponents, $\nu_\perp \zeta / 2 = 0.55 \pm 0.05$, which is again consistent with the results derived from T_{H3}.

Next, we compare the T_{BG} values derived from Eqs. (7.11)-(7.16) to the onset temperatures of T_{H3} in the low frequency limit. We find that for a given θ, the onset of T_{H3} generally occurs at a temperature higher than T_{BG}, suggesting that nonlinear vortex response occurs *above* the Bose-glass transition. Interestingly, we also notice that the onset temperature for T_{H3} is very close to the temperature window within which critical fluctuations dominate and the scaling laws in Eqs. (7.11)-(7.16) apply. Consequently, we may conclude that the high sensitivity of T_{H3} measurements (capable of measuring resistivity $>\sim 10^{-11}\Omega$m)[13] provides an accurate detection of the rapidly increasing critical fluctuations as the temperature approaches a second-order vortex phase transition. In the inset of Figure 7.13 we show the Bose-glass H vs. T phase diagram by using the $T_{BG}(0°)$ values derived from Eqs.(7.11)-(7.13) for different dc magnetic fields. The shaded region indicates the critical regime, and the curve $H_{onset}(T)$ shows the onset of T_{H3} which is clearly within the critical regime of the Bose-glass transition.[6] The results are consistent with those obtained from our previous ac transport measurements on a similar sample.[5,6]

Finally, we consider the third harmonic data for $\theta > \theta_{cr}$. Our previous ac transport studies[5,6] have shown that the vortex phase transition for $\theta >> \theta_{cr}$ resembles that of the vortex-glass transition for samples with random point defects. Thus, the angular dependence of τ is determined by the vortex-glass exponents ν and z[5,6] of the sample. The corresponding scaling relations for $\tau(\theta >> \theta_{cr})$ become

$$\tilde{\tau}(\tilde{f}) = \tau\delta^{\nu(z-1)}, \quad \tilde{f} = f\delta^{-\nu z}, \quad \delta \equiv |1 - T/T_G| \tag{7.18}$$

where T_G is the vortex-glass to vortex-liquid transition temperature. By applying Eq. (7.18) to the τ vs. T curves taken at $\theta = 90°$ and H = 5kOe, we find that $\nu(z\text{-}1) = 1.5 \pm 0.2$ and $\nu z = 2.1 \pm 0.1$, consistent with our previous findings of $\nu \approx 0.7$ and $z \approx 3.0$ for the vortex-glass transition.[5,16,17]

In summary, we have developed a self consistent one-loop model which quantitatively accounts for the third harmonic transmissivity (T_{H3}) data of a $YBa_2Cu_3O_7$ single crystal with parallel columnar defects. The model incorporates a characteristic relaxation time τ which provides direct information about vortex critical fluctuations near the Bose-glass transition, as well as a characteristic crossover current density j which includes the effects of strong pinning at low temperatures. Critical scaling relations are derived and applied to the T_{H3} data for the first time. The results are in excellent agreement with the Bose-glass theory and with our previous ac transport measurements on a similar sample, thus providing a conceptual foundation for investigating critical dynamics of vortex phase transitions with the third harmonic technique.

References

[1] For a review of vortex properties in high temperature superconductors, see G. Blatter et al., Rev. Mod. Phys. (in press).

[2] D. S. Fisher, M. P. A. Fisher, and D. Huse, Phys. Rev. B **43**, 130 (1991).

[3] D. R. Nelson and V. M. Vinokur, Phys. Rev. Lett. **68**, 2398 (1992); Phys. Rev. B **48**, 13060 (1993).

[4] T. Hwa, P. Ledoussal, D. R. Nelson, and V. M. Vinokur, Phys. Rev. L **71,** 3545 (1993).

[5] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, F. Holtzberg, M. Konczykowski, C. C. Tsuei, and C. C. Chi, Physica A **200**, 374 (1993); to appear in Physica C (1994).

[6] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, D. A. Beam, M. Konczykowski, T. A. Tombrello, and F. Holtzberg, Phys. Rev. Lett. **72,** 550 (1994).

[7] C. J. van der Beek, V. B. Geshkenbein, and V. M. Vinokur, Phys. Rev. B **48**, 3393 (1993).

[8] M. Konczykowski, F. Rullieralbenque, E. R. Yacoby, A. Shaulov, Y. Yeshurun, and P. Lejay, Phys. Rev. B **44,** 7167 (1991).

[9] L. Civale, A. D. Marwick, T. K. Worthington, M. A. Kirk, J. R. Thompson, L. Krusin-Elbaum, Y. Sun, J. R. Clem, and F. Holtzberg, Phys. Rev. Lett. **67**, 648 (1991).

[10] W. Gerhaüser, G. Ries, H. W. Neumuller, W. Schmidt, O. Eibl, G. Saemannischenko, and S. Klaumunzer, Phys. Rev. Lett. **68**, 879 (1992).

[11] D. S. Reed, N.-C. Yeh, M. Konczykowski, and A. V. Samoilov, submitted to Phys. Rev. Lett. (1994).

[12] L. Krusin-Elbaum, L. Civale, G. Blatter, A. D. Marwick-Ad, F. Holtzberg, and C. Feild, Phys. Rev. Lett. **72**, 1914 (1994).

[13] A. V. Samoilov and M. Konczykowski, submitted to Phys. Rev. Lett. (1994).

[14] C. J. van der Beek, M. Konczykowski, V.M. Vinokur, T. W. Li, P. H. Kes, and G. W. Crabtree, submitted to Phys. Rev. Lett. (May, 1994).

[15] J. Gilchrist and M. Konczykowski, Physica C **212**, 43 (1993).

[16] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6150 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 413.

[17] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, D. A. Beam, and F. Holtzberg, Phys. Rev. B **49**, 4384 (1994).

[18] N.-C. Yeh, U. Kriplani, W. Jiang, D. S. Reed, D. M. Strayer, J. B. Barner, B. D. Hunt, M. C. Foote, R. P. Vasquez, A. Gupta, and A. Kussmaul, Phys. Rev. B **48**, 9861 (1993).

[19] A. Shaulov and D. Dorman, Appl. Phys. Lett. **53**, 2680 (1988).

[20] M. Konczykowski, F. Holtzberg and P. Lejay, Superc. Sci. Techn. **4**, S331 (1991).

[21] See for example C. Kittel, *Introduction to Solid State Physics*, 6th edn. (John Wiley & Sons, Inc., New York, 1986), Chapter 8.

[22] M. Konczykowski, private communications.

[23] G. Blatter, V. B. Geshkenbein, and A. I. Larkin, Phys. Rev. Lett. **68**, 875 (1992).

[24] R. G. Beck, D. E. Farrell, J. P. Rice, D. M. Ginsberg, and V. G. Kogan, Phys. Rev. Lett. **68**, 1594 (1992).

Chapter 8 : Splayed Glass

The presence of columnar defects has been shown to have a dramatic effect on the pinning of flux lines as evidenced by a dramatically enhanced critical current density[1-3]. As was discussed in Chapter 7, the one-dimensional nature of the parallel columnar defects leads to a broken symmetry between the direction parallel to the columns and perpendicular to the columns. The broken symmetry and strongly enhanced pinning lead to a new Bose-glass phase[4] of the vortex-solid that is characterized by a different set of critical exponents from those found is a sample with only point defects. Recent theoretical work[5] has suggested that the introduction of a slight splay in the angles of the columnar defects could lead to a new "splayed-glass" phase of the vortex-solid that would show significantly enhanced pinning over the Bose-glass system with only parallel columnar defects. The enhancement in pinning is due to an artificial entanglement of the flux lines which is not seen in the Bose-glass system. The flux line entanglement also leads to a much longer relaxation time for the flux lines to relax to the ground state, which could lead to a different universality class than the Bose-glass system with different critical exponents. The slower relaxational dynamics of a splayed-glass system would be evident in an increased dynamic exponent z'.

In order to investigate the existence of the new splayed-glass phase, we investigate a simplified version of the splay of angles of columnar defects. The sample investigated here is a $YBa_2Cu_3O_7$ single crystal with columnar defects at two different angles $\pm 7.5°$ relative to the crystalline c-axis so that all columnar defects will lie in the xz plane as shown in Figure 8.1. In this sample with canted columnar defects, the symmetry is broken in all three (x, y, and z) directions, which, as discussed in Chapter 3, gives rise to three distinct static exponents, ν_x, ν_y, and ν_z. In the case of columnar defects at random orien-

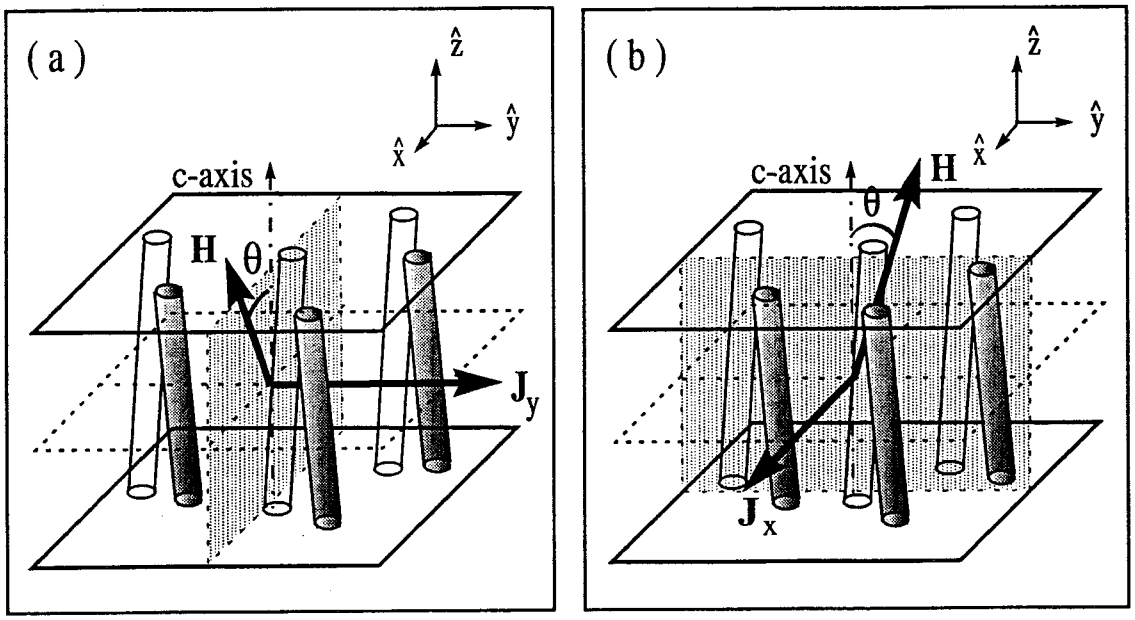


Figure 8.1. Two different experimental configurations for the studies of splayed-glass. The columnar defects are at ±7.5° relative to the crystalline c-axis and on the *xz* plane, and $\hat{c}||\hat{z}$. **(a)** The applied current density $\vec{J}||\hat{y}$, and the magnetic field $\vec{H}$ on the *xz* plane. **(b)** $\vec{J}||\hat{x}$, and $\vec{H}$ on the *yz* plane.

tations relative to the c-axis, the x and y degeneracy is no longer lifted, giving rise to a system with critical phenomena possibly resembling those of the Bose-glass, but with enhanced pinning at low temperatures.

A second-order splayed-glass transition for magnetic fields at small angles relative to the columns is investigated using critical scaling dc transport measurements and the ac impedance measurements developed for the work described in Chapter 5. Using these techniques, the critical exponents and the anisotropic vortex phase diagram are determined for the first time in a system with canted columnar defects. The vortex dynamics of this anisotropic splayed-glass system is also compared with that of the Bose-glass and vortex-glass transitions to see whether the artificial entanglement has in fact led to a new phase with dramatically slower dynamics.

8.1 Experimental Setup

The sample investigated in this work is a $YBa_2Cu_3O_7$ single crystal irradiated twice with a dose of 1.0 GeV Xe ions, corresponding to a total matching field of $B_\varphi = 2\,\mathrm{T}$ and an average separation ~346Å between neighboring columnar defects. For the first irradiation, the beam was aligned at +7.5° from the crystal c-axis, and for the second irradiation, at -7.5°. The crystal is twinned with an average twin boundary separation of ~10μm and sample dimensions of 0.7mm×0.4mm×15μm with the shortest dimension along the crystal c-axis. The zero field superconducting transition temperature is T_c = 92.08K, with a resistive transition width of ~0.15K and a normal state resistivity of 65$\mu\Omega$-cm. Both dc and ac electrical transport measurements are performed using the standard four probe technique. The dc measurements of the electric field (E) versus current density (J) isotherms are studied as a function of the applied magnetic fields (H) and the magnetic field orientation (θ) relative to the sample c-axis. As shown in Figure 8.1, two experimental configurations are used in the investigations: one with $\vec{J}||\hat{y}$ and $\vec{H}$ on the xz plane, the other with $\vec{J}||\hat{x}$ and $\vec{H}$ on the yz plane, where $\hat{z}$ is parallel to the sample c-axis. The same configurations are used in the measurements of ac impedance isotherms as a function of the applied current frequency (f). The ac current is held constant at 50μA and is swept in frequency from 100 Hz to 10 MHz. Details of the ac measurement techniques and the calibration procedures for isotherms of the frequency dependent impedance have been given in Chapter 5. All isotherms are measured with a temperature stability of ±0.005K.

8.2 Experimental Results

8.2.1 $|\theta|$<10° – Splayed-Glass Transition

Using the scaling relations derived in Chapter 3 for an anisotropic system, we assume that a second-order splayed-glass transition temperature $T_{SG}(H,\theta)$ exists for a given magnetic field H and a small angle θ between $\vec{H}$ and $\hat{c}$. Three vortex correlation lengths

$\xi_i(T) = \xi_{i0}\left|1-\left(T/T_{SG}\right)\right|^{-\nu_i}$, with the coordinates i=x,y,z defined in Figure 8.1, are needed to describe the phase transition at a given T_{SG}. For the configuration defined in Figure 8.1(a), the relevant vortex correlation lengths are ξ_x and ξ_z because $\vec{J}\|\hat{y}$. Thus, for given values of H and θ, the ac resistivity $\rho(T,f,J)$ follows the scaling relation:

$$\rho(T,f,J\to 0) = \delta^{(\nu_x z'-\nu_z)}\tilde{\rho}_{\pm}\left(\tilde{f}\right), \qquad \tilde{f} \equiv f\delta^{-\nu_x z'}, \qquad \delta \equiv \left|1-T/T_{SG}\right| \tag{8.1}$$

where $\tilde{\rho}_+$ and $\tilde{\rho}_-$ denote the scaling functions for $T>T_{SG}$ and $T<T_{SG}$, respectively. Similarly, the electric field (E) versus current density (J) isotherms in the dc limit ($f \to 0$) follow the scaling expression:

$$E(T,f\to 0,J) = \delta^{(\nu_x z'-\nu_z)}\tilde{E}_{\pm}\left(\tilde{J}\right), \qquad \tilde{J} \equiv \left(J/T\right)\delta^{-(\nu_x+\nu_z)}, \qquad \tilde{E}_{\pm} = J\tilde{\rho}_{\pm} \tag{8.2}$$

Using Eq. (8.1), the isotherms of $\rho(f)$ vs. f data for field orientations close to the columnar defects ($\theta \lesssim 10°$, θ being the angle between the applied field and the sample c-axis) can be scaled into two universal functions by using the exponents $\left(\nu_x z' - \nu_z\right) = 2.50 \pm 0.05$ and $\nu_x z' = 3.68 \pm 0.05$, as shown in Figure 8.2 for the data taken at H=5.0 kOe and θ =0°. The splayed-glass transition temperatures $T_{SG}(H,\theta)$ are also determined. Similarly, the E vs. J isotherms for $\theta \le 10°$ can be scaled into universal functions with the exponents $\nu_x z' - \nu_z = 2.50 \pm 0.05$ and $\left(\nu_x + \nu_z\right) = 2.30 \pm 0.10$ by using Eq. (8.2), as shown in Figure 8.3(a) and (b) for the representative data taken at H=5 kOe, θ=2.5° and H=10 kOe, θ=0°, respectively. Thus, the critical exponents $\nu_x(\equiv \nu_\perp) = 1.1 \pm 0.1$, $\nu_z(\equiv \nu_\parallel) = 1.2 \pm 0.1$, and $z' = 3.8 \pm 0.4$ are determined self consistently from the above analysis.

The presence of twin boundaries in the sample introduces some competing effects between these twin boundaries and the splayed columnar defects. Since the average separation between columnar defects is ~346Å, much smaller than the average twin boundary separation ~10 μm, one expects the columnar defects to be the dominant disorder until the

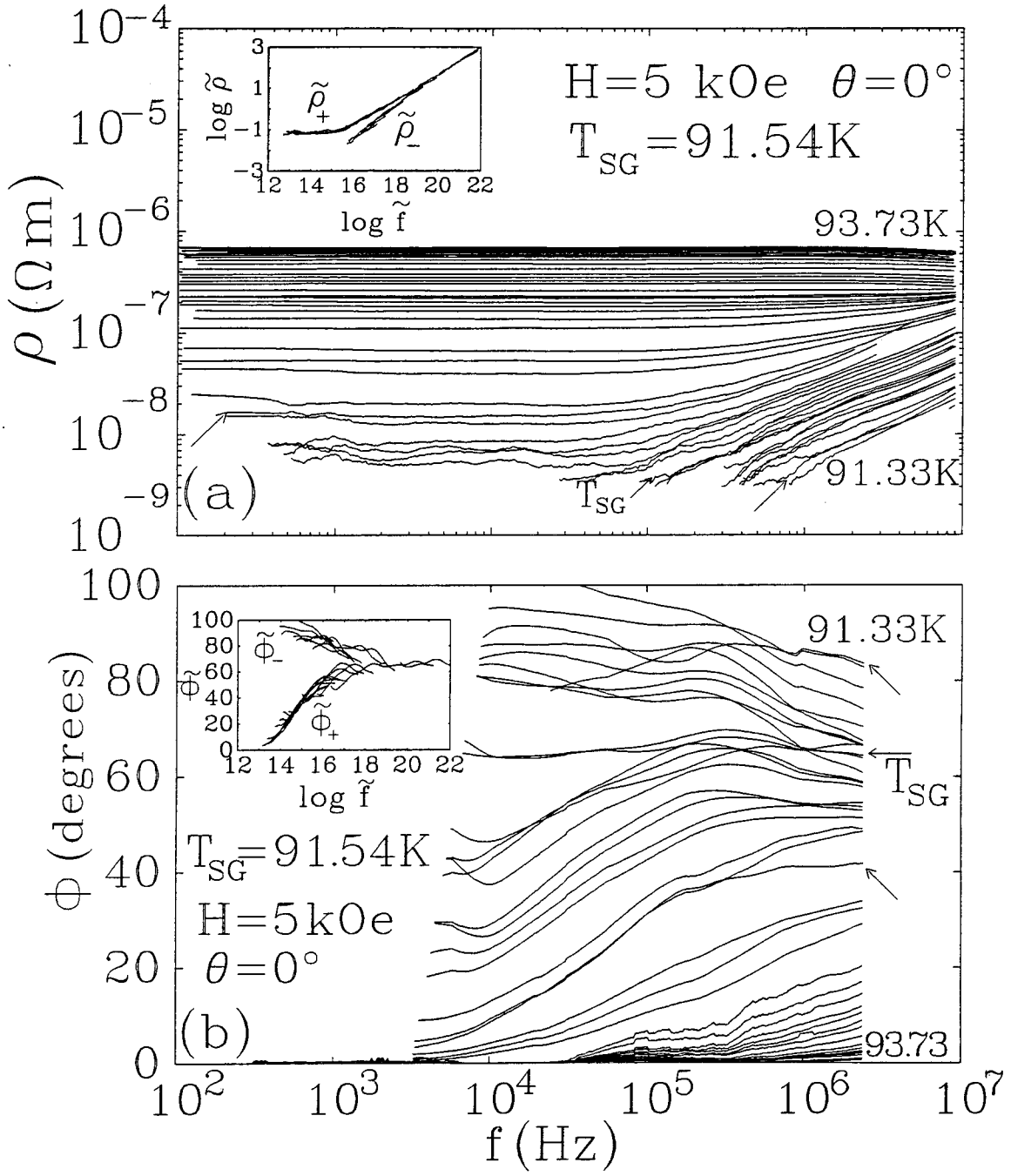


Figure 8.2. **(a)** Amplitude of the ac resistivity ρ vs. frequency (f) isotherms taken at H=5.0 kOe and θ=0° and **(b)** the corresponding phase of the ac resistivity. The arrows indicate the isotherms in the critical regime which are scaled into universal functions $\tilde{\rho}_{\pm}(\tilde{f})$ and $\tilde{\Phi}_{\pm}(\tilde{f})$ in the inset by using Eq. (8.1). The temperature increment in the critical regime is $\Delta T \approx$ 0.03K.

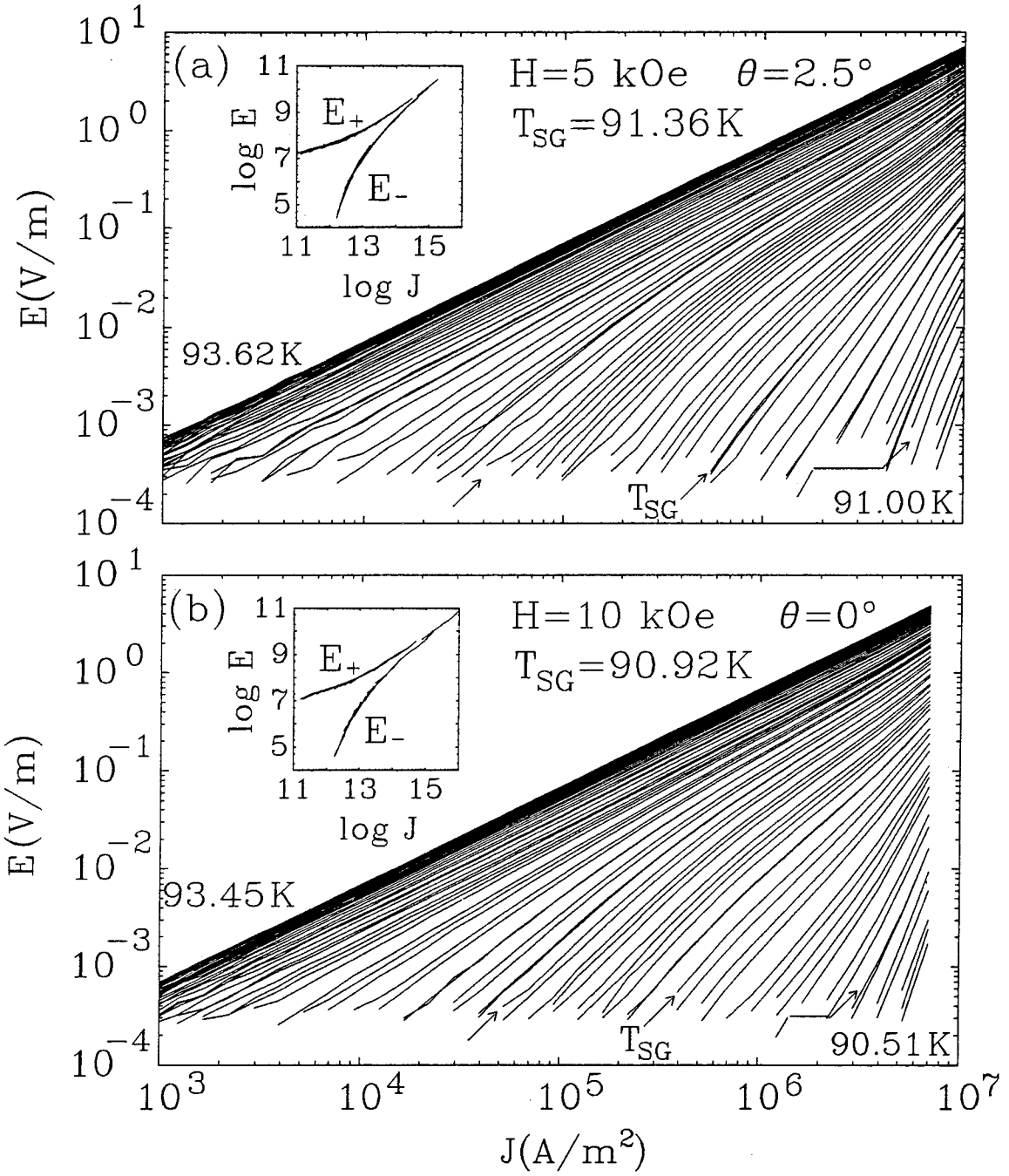


Figure 8.3. The E vs. J isotherms of the splayed-glass system for **(a)** H=5 kOe, θ=2.5°; and **(b)** H=10 kOe, θ=0°. Note that the critical exponents ν_i (i=x,y,z) and z' have been used for all data taken at θ<10° to yield the same universal functions by using Eq. (8.2).

vortex system approaches a three-dimensional vortex phase transition where two-dimensional defects such as twin boundaries eventually impose the finite size effect[6-8] as discussed in Chapter 3.4. Thus, in performing critical scaling of the E vs. J isotherms, one has to take into account the finite size imposed current density J_ℓ.[6] Namely, the scaling relation in Eq. (8.2) is only applicable for $J > J_\ell(T) \sim 10^5\,\mathrm{A/m^2}$ with $\ell \sim 10\mu\mathrm{m}$.[6] The scaling functions shown in the insets of Figure 8.3 and Figure 8.4 are obtained by taking such an effect into consideration. Because of the limited current range in which the critical scaling analysis can be applied, it is difficult to conclusively identify the transition temperatures from dc measurements alone. However, no such limitation occurs in the collapsing of the ρ vs. f isotherms so the ac impedance measurements can be used to accurately identify the transition temperature. With the transition temperatures known, the dc E vs. J isotherms can be collapsed accurately to give the information needed to obtain the critical exponents. Even without the limitations of finite size effects, the dc measurements alone would not have been sufficient to identify all of the critical exponents ν_x, ν_y, and z. The scaling analysis of the E vs. J isotherms give only two equations with three unknowns, and only by having an additional experimental technique can all of the critical exponents be obtained. The need for this additional information underscores the importance of developing the various experimental techniques discussed in Chapters 5-7.

The procedure is repeated for the other experimental configuration indicated in Figure 8.1(b), with $\vec{J} \| \hat{x}$ and $\vec{H}$ on the yz plane. By applying both Eqs. (8.1) and (8.2) to the ac and dc transport data, with ν_x replaced by ν_y, we obtain the value for $\nu_y (\equiv \nu'_\perp) = 1.4 \pm 0.3$. In addition, the values for $\nu_z = 1.2 \pm 0.2$ and $z' = 3.7 \pm 0.6$, as well as the transition temperatures $T_{SG}(H, \theta)$, are consistent with the results derived from the configuration in Figure 8.1(a) within experimental accuracy. We point out that the errors given in the critical exponents are combined statistical errors from both ac and dc meas-

urements and for all H and θ values. It should be emphasized that the values of the exponents are correlated in such a way that they are always different from each other, with the best fits given by the mean. Thus, ν_y is found to be always larger than ν_x for all data investigated, consistent with stronger impedance for vortex motion along the direction $\hat{y}$ which is transverse to the canted columns. Furthermore, the significantly larger dynamic exponent $z' \approx 3.8$ as compared to that for the Bose-glass transition ($z' \approx 2.2$) indicates a slower vortex relaxation, consistent with the higher degrees of disorder in splayed-glass.

8.2.2 80°<θ<90° – Vortex-Glass Transition

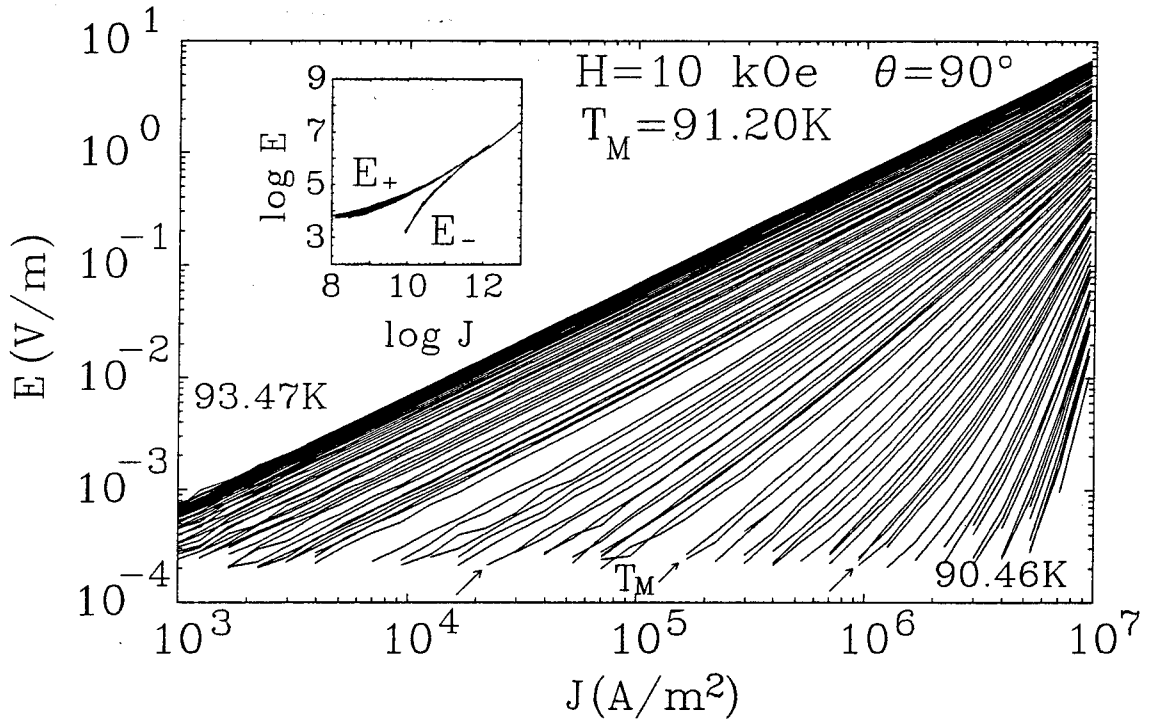


Figure 8.4. The E vs. J isotherms for H=10kOe and θ=90°. The same critical exponents ν and z found for the vortex-glass system are used to collapse the isotherms into the scaling functions shown in the inset.

As we rotate the magnetic field farther away from the columnar defects, we find very interesting anisotropic features in the vortex dynamics. For very large angles, $80° < \theta \leq 90°$, the critical exponents ($\nu = 0.70 \pm 0.05$ and $z = 3.0 \pm 0.3$) derived from the

ac and dc transport data are consistent with those for the vortex-glass transition,[9] as exemplified in Figure 8.4 for H=10 kOe and θ=90°. This finding is consistent with the fact that the intersections between columnar defects and perpendicular flux lines are at equivalently single points. Therefore, the vortex phase transition is expected to belong to the same universality class as that of the vortex-glass transition for systems with random point defects.[4,9,10] We note that similar phenomena have also been observed in the Bose-glass system for samples with parallel c-axis columnar defects.[9]

8.2.3 10°<θ<80° – Intermediate Regime

Despite clear experimental evidence of second-order phase transitions for magnetic fields oriented at $0 \leq \theta <\sim 10°$ and $80° <\sim \theta \leq 90°$ from both dc and ac measurements, the vortex transport data for intermediate angles ($10° < \theta < 80°$) show no signs of a vor-

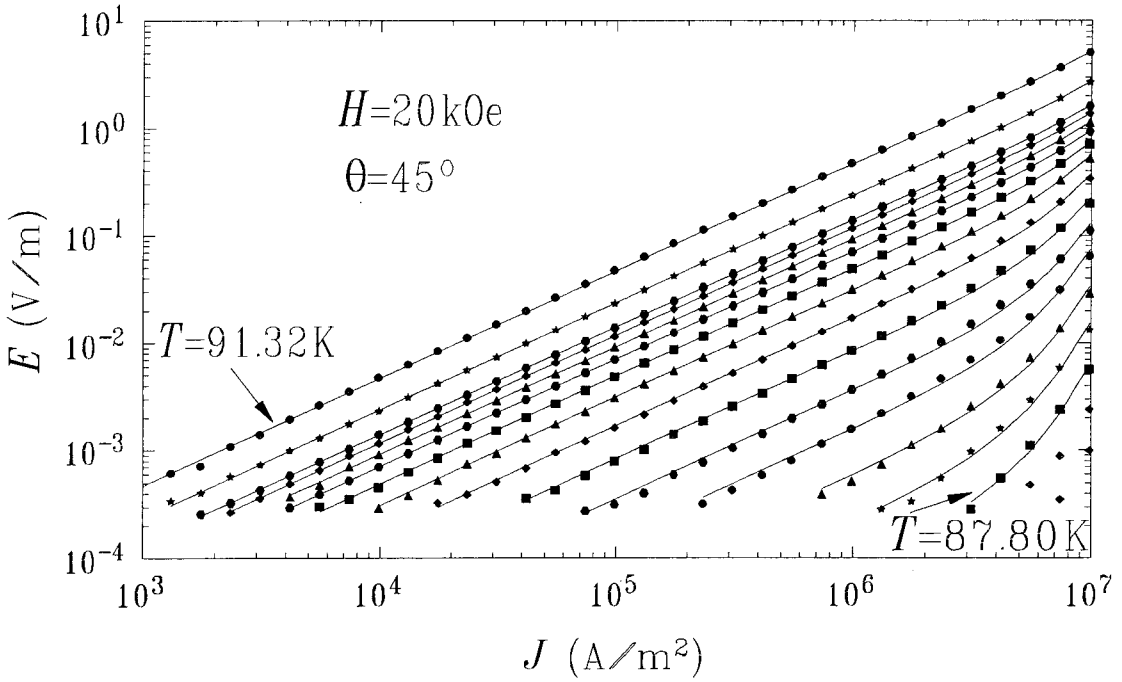


Figure 8.5. E vs. J isotherms for H=20kOe and θ=45°. The solid line is a fit using Eq. (8.3). This function fits the data over the full current range only for angles 10°<θ<80°. At other angles, U becomes current dependent so that Eq. (8.3) can be used to fit the data only in a narrow current range $J >\sim 5 \times 10^5$ A/ m^2.

tex-solid to vortex-liquid phase transition at all temperatures of our investigation. In other words, neither the E vs. J isotherms nor the ρ vs. f isotherms can be scaled into universal functions with any reasonable exponents. Instead, as shown in Figure 8.5, the dc transport data can be described by the conventional theory of thermally activated flux flow (TAFF),[11] so that for given values of H and θ, the formula

$$E = \rho_0 \exp\left[-\frac{U(T,H)}{k_B T}\right] \sinh\left[\frac{J\alpha}{k_B T}\right] \tag{8.3}$$

applies to the data, where ρ_0 is a resistivity coefficient, $\alpha=HV_cL$ with H, V_c, and L as defined for Eq. (2.1), and the energy barrier for vortex motion is given by $U(T,H)$. For the TAFF behavior found in the intermediate angles, ($10°<\theta<80°$), $U(T,H)$ is current independent and therefore finite for all current densities, in contrast to the splayed-glass and vortex-glass phases where $U \to \infty$ for $J \to 0$.[5,12] By fitting the isotherms as shown in Figure 8.5, we obtain $U(T)$ for a given magnetic field and angle. By repeating this fitting for various angles θ, we obtain the U vs. T data shown in Figure 8.6 for H=30 kOe. For angles $|\theta|<10°$ and $80°<\theta<90°$, the $U(T)$ is current dependent, but in a narrow current range at large currents, Eq. (8.3) can be used to find a characteristic $U(T)$ which is also shown in Figure 8.6.

In Figure 8.6, we can identify $U(T_{SG})$ and $U(T_M)$ for the data taken at $\theta \le 10°$ and $80° < \theta \le 90°$, respectively, and we note that these $U(T)$ values are found to satisfy the empirical rules, $U(T_{SG})=7.4k_BT_{SG}$ and $U(T_M)= 7.4k_BT_M$. Consequently, by applying the same empirical rule $U=7.4k_BT$ to the U vs. T data at intermediate angles shown in Figure 8.6, we obtain the characteristic temperatures $T_{cr}(H, \theta)$ for $10°<\theta<80°$. These characteristic temperatures are shown along with the transition temperatures T_M and T_{SG} in Figure 8.7.

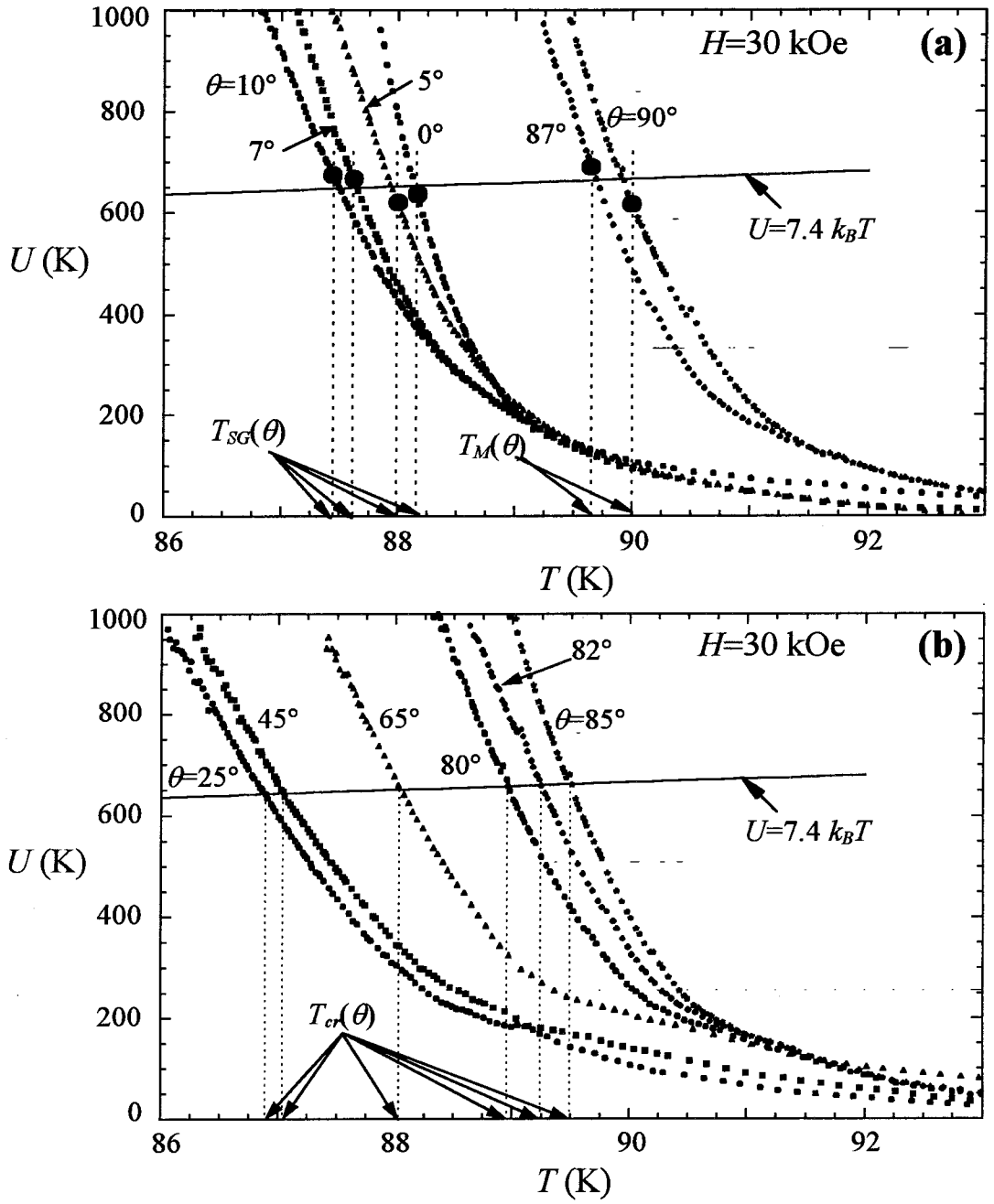


Figure 8.6. **(a)**The temperature dependence of the characteristic energy barrier $U(T, H, \theta)$ derived from the data taken at $\theta \leq 10°$ and $\theta > 80°$ and H=30kOe. Note that the phase transition temperatures roughly correspond to the intersection between the $U(T)$ curve and the line U=7.4k_BT. **(b)** The temperature dependence of the characteristic energy barrier $U(T, H, \theta)$ derived from the data taken at $10° < \theta < 80°$ and H=30 kOe. Note that the temperatures where the experimentally determined $U(T)$ curves and the line U=7.4k_B T intersect determine the characteristic temperatures $T_{cr}(H, \theta)$ for the angles $10° < \theta < 80°$ shown in Figure 8.7.

8.3 Discussion

The anisotropic vortex phase diagram (T vs. θ) for all angles and different magnetic fields at H=5, 10, 15, 20 and 30 kOe is shown in Figure 8.7. We note that the angular dependence is similar to, but slightly different from that found in the Bose-glass transi-

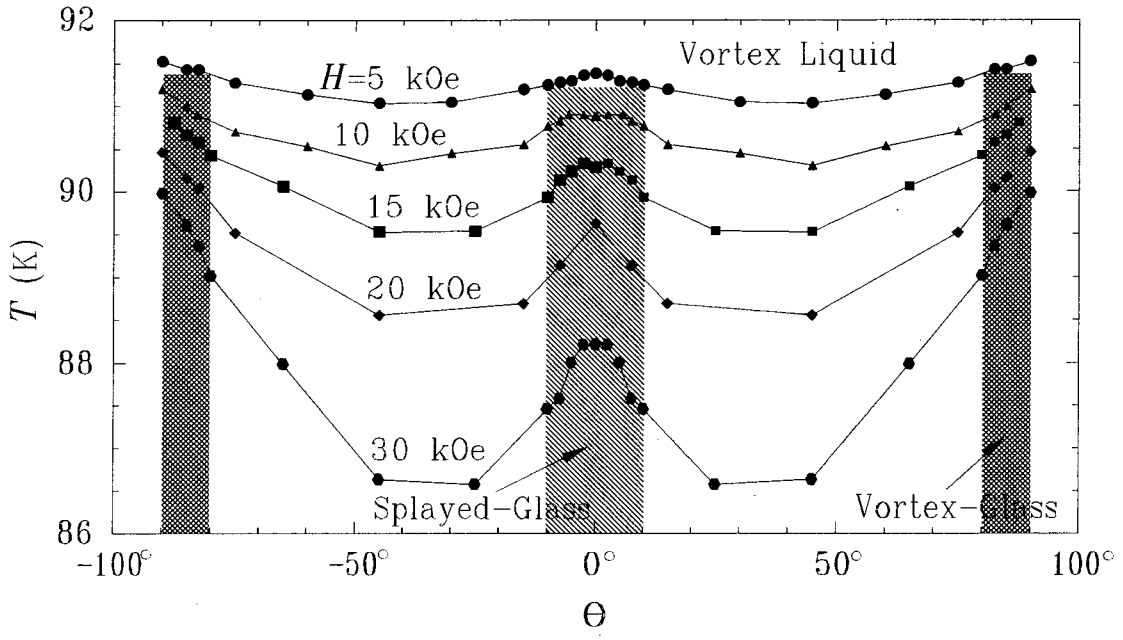


Figure 8.7. The anisotropic vortex phase diagram T vs. θ for the splayed-glass system at various magnetic fields H=5, 10, 15, 20 and 30 kOe.

tion for $YBa_2Cu_3O_7$ single crystals with parallel columnar defects.[9] The general feature of maxima of the transition temperatures for fields applied parallel to the columns and fields in the ab plane still remains. However, the angle beyond which the splayed-glass transition breaks down is θ~10°, considerably smaller than the angle (θ~30°) found in the Bose-glass transition,[9] suggesting enhanced confinement of vortices due to the presence of canted columns at ±7.5°. Similarly, the vortex-glass transition for the splayed-glass system is only observed for magnetic fields at θ=90° ± 10°, in contrast to the Bose-glass system where it is found at θ=90° ± 30°.

To facilitate better understanding of the effects of transverse fields ($H_\perp \equiv H\sin(\theta)$) on the vortex phase transition, the following scaling transformation[13] is performed to remove effects of the intrinsic anisotropy in $YBa_2Cu_3O_7$, so that θ is converted to an effective angle $\tilde{\theta}$ for an isotropic superconductor:

$$\sin\tilde{\theta} \equiv \frac{\tilde{H}_\perp}{\tilde{H}} = \frac{\varepsilon \sin\theta}{\sqrt{\cos^2\theta + \varepsilon^2 \sin^2\theta}}, \qquad \tilde{H} \equiv H\sqrt{\cos^2\theta + \varepsilon^2 \sin^2\theta} \tag{8.4}$$

where $\varepsilon^{-2} \approx 60$ is the mass anisotropy for $YBa_2Cu_3O_7$.[14] Thus, the effective transverse field $\tilde{H}_\perp$ in the isotropic frame is given by $\tilde{H}\sin\tilde{\theta}$. We can also rescale the transition temperatures into the isotropic frame by considering the relation:

$$H_{SG}(T,\theta) = H_{SG0}(\theta)\left|1 - \frac{T}{T_{c0}}\right|^{2\nu_0} \tag{8.5}$$

where T_{c0} is the zero field transition temperature and ν_0 is the zero field critical exponent. The temperature in Eq. (8.5) can be rescaled into the isotropic frame $T \to \tilde{T}$ by rescaling the coefficient $H_{SG0}(\theta) \to \tilde{H}_{SG0}(\theta)$ at a fixed field H_{SG}(T), giving the relation

$$\tilde{T} = T_{c0} + T_{c0}\left(\frac{H_{SG}}{\tilde{H}_{SG0}}\right)^{1/2\nu_0} = T_{c0} + \frac{T - T_{c0}}{\left(\sqrt{\cos^2(\theta) + \varepsilon^2 \sin^2(\theta)}\right)^{1/(2\nu_0)}}. \tag{8.6}$$

We find the value $\nu_0 = 0.5 \pm 0.05$ by fitting Eq. (8.5) to the H_{SG} vs. T phase boundary shown in Figure 8.9. After rescaling the temperature and field according to Eqs. (8.4) and (8.6), the resulting $\tilde{T}$ vs. $\left(\tilde{H}_\perp/\tilde{H}\right)$ phase diagram is depicted in Figure 8.8 for different magnetic fields. We note that for all fields the splayed-glass temperatures $\tilde{T}_{SG}(H,\tilde{\theta})$ follow the empirical law

$$\left[\tilde{T}_{SG}(H,\tilde{\theta}) - \tilde{T}_{SG0}(H)\right] \propto \left(\sin\tilde{\theta} - \sin\tilde{\theta}_{cr}\right)^{\alpha}, \tag{8.7}$$

where $\tilde{T}_{SG0}(H) = \tilde{T}_{SG}(H, \theta = \theta_{cr})$, $\alpha = 0.54 \pm 0.07$, and $\sin\tilde{\theta}_{cr} = 0.004 \pm 0.002$. This value of $\sin\tilde{\theta}_{cr}$ corresponds to a value of $\theta_{cr} = 1.8° \pm 0.9°$. From Figure 8.8, we see

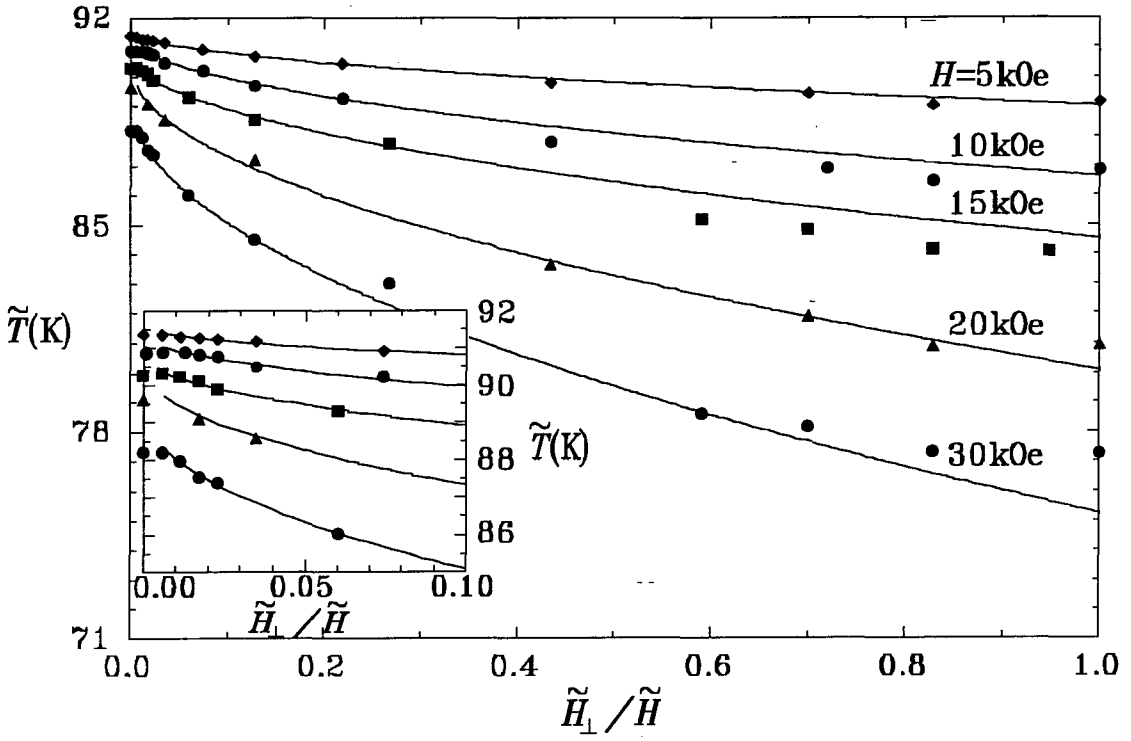


Figure 8.8. The scaled phase diagram $\tilde{T}$ vs. $\left(\tilde{H}_\perp / \tilde{H}\right)$ obtained by using Eq. (8.4) and (8.6) for different magnetic fields H. The solid lines represent the power law dependence in Eq. (8.7) with $\sin\left(\tilde{\theta}_{cr}\right) = 0.004 \pm 0.002$, and $\alpha = 0.54 \pm 0.07$. The inset shows an expanded view of the low angle data.

very good agreement between the experimental data and Eq. (8.7) except for very large angles and very small angles $\tilde{\theta} < \tilde{\theta}_{cr}$. A quantitative understanding of the angular scaling behavior awaits further theoretical investigation.

In Figure 8.9, the H vs. T vortex phase diagram for $\vec{H} \| \hat{c}$ is shown for three $YBa_2Cu_3O_7$ single crystals with point, parallel and canted columnar defects. It is clear that in the reduced temperature (T/T_c) scale, both the splayed-glass and Bose-glass phase boundaries ($H_{SG}(T)$ and $H_{BG}(T)$) for samples with columnar defects lie significantly above that of the vortex-glass melting transition (H_M) for samples with random point defects, indicating significant enhancement of pinning due to the presence of columnar defects.

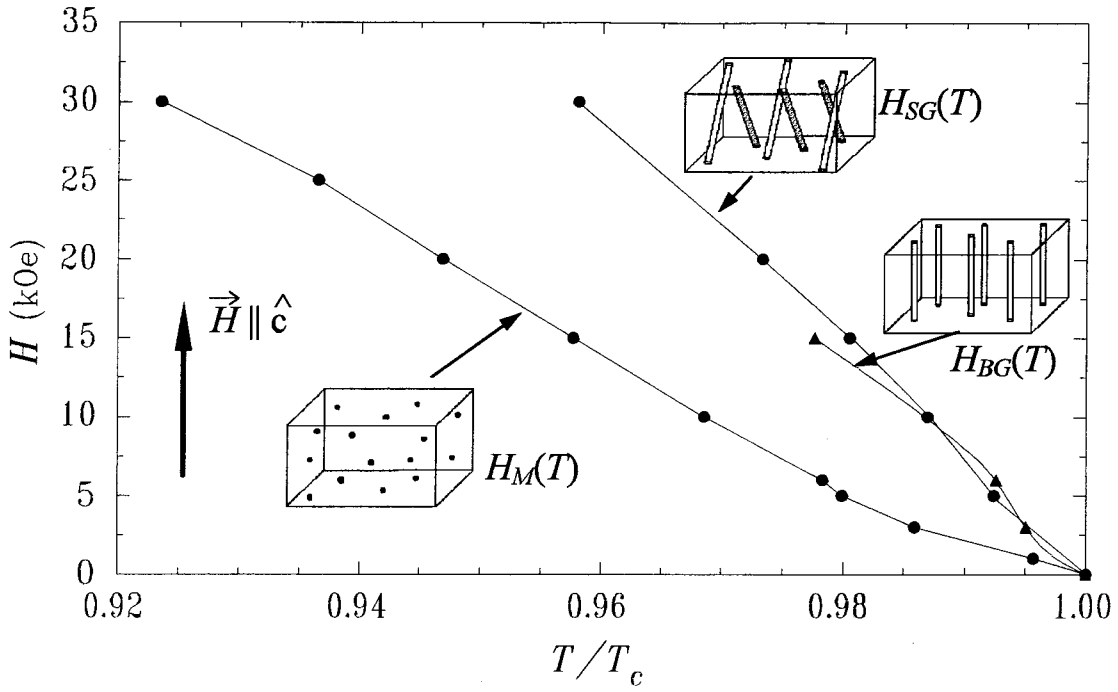


Figure 8.9. Comparison of the H vs. (T/T_c) vortex phase diagram for the vortex-glass (H_M), Bose-glass (H_{BG}) and splayed-glass (H_{SG}) phase transitions in $YBa_2Cu_3O_7$ single crystals with point, parallel, and canted columnar defects. Note that $\vec{H}||\hat{c}$.

In summary, a second-order splayed-glass to vortex-liquid transition is manifested by the critical scaling of vortex transport data on a $YBa_2Cu_3O_7$ single crystal with canted columnar defects. The anisotropic vortex phase diagram and critical exponents $(\nu_\parallel, \nu_\perp, \nu'_\perp, z')$ are determined for the first time. In comparison with the parallel columnar defects in the Bose-glass system, the canted columns in splayed-glass result in stronger disorder which impedes vortex motion, thereby slowing down the critical dynamics near the vortex phase transition.

References

[1] L. Civale, A. D. Marwick, T. K. Worthington, M. A. Kirk, J. R. Thompson, L. Krusin-Elbaum, Y. Sun, J. R. Clem, and F. Holtzberg, Phys. Rev. Lett. **67**, 648 (1991).

[2] M. Konczykowski, F. Rullieralbenque, E. R. Yacoby, A. Shaulov, Y. Yeshurun, and P. Lejay, Phys. Rev. B **44,** 7167 (1991).

[3] R. C. Budhani, M. Suenaga, and S. H. Liou, Phys. Rev. Lett. **69**, 3816 (1992).

[4] D. R. Nelson and V. M. Vinokur, Phys. Rev. Lett. **68**, 2398 (1992); Phys. Rev. B **48**, 13060 (1993).

[5] T. Hwa, P. Ledoussal, D. R. Nelson, and V. M. Vinokur, Phys. Rev. L **71,** 3545 (1993); T. Hwa, D. R. Nelson, and V. M. Vinokur, Phys. Rev. B **48**, 1167 (1993).

[6] N.-C. Yeh, D. S. Reed, W. Jiang, U. Kriplani, C. C. Tsuei, C. C. Chi, and F. Holtzberg, Phys. Rev. Lett. **71**, 4043 (1993).

[7] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6146 (1993).

[8] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, T. A. Tombrello, A. P. Rice, and F. Holtzberg, Phys. Rev. B **47**, 8308 (1993).

[9] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, D. A. Beam, M. Konczykowski, T. A. Tombrello, and F. Holtzberg, Phys. Rev. Lett. **72,** 550 (1994).

[10] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, F. Holtzberg, M. Konczykowski, C. C. Tsuei, and C. C. Chi, Physica A **200**, 374 (1993); N.-C. Yeh *et al.*, to appear in Physica C (1994).

[11] P. H. Kes, J. Aarts, J. Vandenberg, C. J. Van der Beek, and J. A. Mydosh., Supercond. Sci. Technol. **1**, 242 (1989).

[12] M. V. Feigelman, V. B Geshkenbein, A. I. Larkin, and V. M. Vinokur, Phys. Rev. Lett. **63**, 2303 (1989).

[13] G. Blatter, V. B. Geshkenbein, and A. I. Larkin, Phys. Rev. Lett. **68**, 875 (1992).

[14] R. G. Beck, D. E. Farrell, J. P. Rice, D. M. Ginsberg, and V. G. Kogan, Phys. Rev. Lett. **68**, 1594 (1992).

Chapter 9 : Conclusion

This thesis has focused on using frequency dependent measurements to investigate the vortex dynamics near the vortex-solid to vortex-liquid phase transition in high temperature superconductors. Three different experimental techniques have been developed to provide more insight into the nature of the vortex dynamics and to verify the self consistency of the results obtained. By using these new experimental techniques, we are able to obtain conclusive evidence for the existence of second-order vortex-solid (including vortex-glass, Bose-glass, and splayed-glass) to vortex-liquid phase transitions. In addition, we have developed new methods of finding the critical exponents associated with the second-order vortex phase transition. These new experimental techniques have not only led to a better understanding of the vortex systems, but also provided versatile new tools applicable to future investigations of the vortex dynamics in superconductors, and perhaps even of the spin dynamics in magnetic materials.

9.1 Experimental Goal

This research works toward developing a better understanding of the vortex dynamics near a vortex-solid to vortex-liquid phase transition. Because the vortex-solid to vortex-liquid phase transition is expected to be second-order in most high temperature superconductors, it is convenient to investigate the vortex system in terms of the theory of critical phenomena. Before this work, most investigations into the critical phenomena associated with this transition involved measurements of the dc current-voltage characteristics. However, as discussed in Chapter 3, both the finite size effects and finite critical currents found in dc transport measurements can limit the current range available for the critical scaling analysis, thereby making the scaling analysis ambiguous. On the other

hand, frequency dependent measurements of the resistivity have been shown to provide a more conclusive demonstration of the scaling analysis. Additionally, by using the different experimental techniques developed in this work, a more stringent test of the universality of the critical exponents can be obtained.

9.2 Experimental Approach

The ac impedance measurement system discussed in Chapter 5 provides a direct measure of the bulk ac impedance of the superconductor. Although this method is conceptually simple (just pass a uniform ac current through the sample and measure the voltage), it is technically quite difficult. Problems of low sample signal, significant contact resistance, cable length, stray capacitive effects, and distortions from electronic components all have to be controlled and compensated with detailed calibrations. The system developed is able to measure impedance over a frequency range of 10^2Hz to 10^7Hz with a sensitivity of 10nV.

The differential two-coil magnetic susceptibility system described in Chapter 6 provides a sensitive way to measure the susceptibility of a superconducting sample over a broad frequency range. This technique presents many of the same experimental challenges associated with measuring small signal levels over a broad frequency range as are found in the ac impedance measurements. The system developed is able to measure the susceptibility over a frequency range of 1kHz to 2MHz. The experiments performed with this system are the first measurements that apply critical scaling analysis to magnetic susceptibility. In developing the analysis, we also presented the first correct scaling relations to describe the ac magnetic susceptibility.

The Hall probe magnetometer discussed in Chapter 7 was developed by Dr. M. Konczykowski at Ecole Polytechnique. The system provides a very accurate measure of

the fundamental and third harmonic transmissivity of the superconducting sample at low frequencies. The frequency range used is 0.79Hz to 2kHz. Although the third harmonic technique provides a very sensitive probe into the nonlinear characteristics, no accurate analysis technique had been developed for the third harmonic measurements before this work. We developed an analysis technique that provides information about a characteristic time τ and a characteristic crossover current j. From these parameters, we are able to present the first critical scaling analysis of the third harmonic transmissivity measurements.

The three frequency dependent measurement systems are versatile tools for investigating the vortex dynamics of various systems. As discussed in Chapter 6.4, measurements of resistivity and susceptibility are connected through electrodynamic theory, though the details of this connection are rather complicated and not fully known. The susceptibility and impedance measurements are complementary in other important ways as well. Susceptibility measurements are non-contact measurements, so that the difficulties of making electrical contacts can be avoided. Many samples that are too small for electrical contacts can still be studied using magnetic susceptibility. However, larger samples including films are better suited to the impedance measurements because larger coils would be needed to accommodate large samples, giving rise to an increase in the stray magnetic field and larger sensitivity to external noise. Perhaps an even more important complementary feature of these two techniques is that impedance measurements measure signals primarily above the phase transition in the vortex-liquid state, while susceptibility measurements measure signals primarily below the phase transition in the vortex-solid state. As a result, a combination of both techniques is required to fully explore the vortex dynamics at temperatures away from the vortex-solid to vortex-liquid transition.

The transmissivity (T_H) measured by the Hall probe is directly related to the susceptibility (χ) measured by the two-coil system through the relation $T_H = 1 + \chi$. However, the two systems are complementary in the frequency range over which measurements can be performed. As described in Chapter 7, the Hall probe's sensitivity is independent of frequency so very low frequency measurements are possible (to 0.79 Hz), but at high frequency, the inductive pickup by the electrical contacts to the Hall probe dominate the Hall voltage signal. This inductive pickup creates a background that becomes larger than the Hall voltage signal at frequencies >~10kHz. On the other hand, as described in Chapter 6, the sensitivity of the two-coil susceptibility system is proportional to frequency, which limits the low frequency signal to ~1kHz, but measurements can be made up to frequencies <~2MHz. The frequency independent response of the Hall probe also makes it better suited to the third harmonic measurements presented in Chapter 7, although in principle third harmonic measurements could also be performed using the two-coil system.

9.3 Experimental Results

Because the vortex-solid to vortex-liquid phase transition in the presence of quenched disorder is expected to be second-order in most high temperature superconductors, it is convenient to investigate the vortex dynamics near the phase transition in terms of the theory of critical phenomena. In Chapter 3, the scaling relations for various experimental systems are derived in terms of the transition temperature $T_M(H)$, the static critical exponent ν, and the dynamic critical exponent z. Although scaling relations have already been derived for the ac impedance (ρ) and susceptibility (χ) in previous theoretical work,[1] the scaling relation for ac magnetic susceptibility was given incorrectly for the superconducting system. Our work is the first to provide the correct scaling relation for

the magnetic susceptibility[2,3] as derived in Chapter 6. We also provide the first experimental demonstration of critical scaling of the ac magnetic susceptibility,[3] as well as the first verification of the critical scaling of both the amplitude and phase of the ac impedance.[4] Additionally, we are the first to derive and verify the critical scaling relations for a characteristic relaxation time (τ) found from analyzing the third harmonic transmissivity experiments[5] described in Chapter 7. In the case of a vortex-glass transition at the temperature $T_M(H)$, the scaling relations for each of these experiments is given by

$$\begin{aligned} \rho &= \tilde{\rho}_{\pm}(\tilde{f})\delta^{\nu(z-1)} \\ \chi &= \tilde{\chi}_{\pm}(\tilde{f})\delta^{\nu/2} \\ \tau &= \tilde{\tau}_{\pm}(\tilde{f})\delta^{-\nu(z-1)} \\ \tilde{f} &\equiv f\delta^{-\nu z} \end{aligned} \tag{9.1}$$

where $\delta \equiv |1 - T/T_M|$, ν is the static exponent, and z is the dynamic exponent. Similarly, scaling relations are derived for anisotropic systems and applied to a Bose-glass transition in samples with parallel columnar defects[6], and a splayed-glass transition in samples with canted columnar defects.[7]

An important point in using the critical scaling analysis to investigate a phase transition is that the critical exponents must show universality for all experimental techniques and for all magnetic fields that do not alter the fundamental nature of the phase transition. By developing the ac transport measurements and ac magnetic susceptibility measurements, we have for the first time used three different experimental techniques to verify the universality of the vortex-glass transition.[2-4,8] This is the first conclusive evidence for the existence of the second-order vortex-glass to vortex-liquid phase transition and confirms the values of the critical exponents ν=0.67±0.05 and z=3.0±0.1 found using dc current-voltage measurements.

For the case of samples with parallel columnar defects, scaling analysis of the third harmonic transmissivity gives critical exponents[5] $\nu_\perp$=0.9±0.2, $\zeta \equiv \nu_\parallel/\nu_\perp$=1.2±0.2, and z'=2.3±0.3 in excellent agreement with our other measurements performed with the ac impedance measurement system.[9] The availability of different experimental techniques to verify the critical exponents provides strong evidence that the values obtained are correct and that the values are truly universal as required at a second-order phase transition.

There are some limitations to the scaling analysis due to finite size effects[8] as discussed in Chapter 3. These effects arise most dramatically in dc current-voltage measurements in which the low current data is limited by finite size effect and the high current data is limited by the flux flow crossover or ohmic heating from the contact resistance. As was seen in the measurements presented in Chapter 8, with only a limited current range over which the scaling analysis is valid, it is difficult to properly interpret the results. However, the frequency dependent measurements are not limited by such effects. Over the frequency range of 10^2 Hz to 10^7 Hz, no finite size limitations are observed, and with the small driving currents used, there is no danger of crossing into the flux flow regime. Therefore, all of the data over five decades in frequency can be used in the scaling analysis to accurately determine the transition temperatures and critical exponents. From this analysis, we are able to identify the critical exponents for the splayed-glass transition of[7] ν_x=1.1±0.1, ν_y=1.4±0.3, ν_z=1.2±0.1, and z'=3.8±0.4. The larger dynamic exponent z' found in the splayed-glass system as compared to the Bose-glass system suggests that the increased disorder caused by the canted columnar defects leads to slower relaxation rates for the flux lines.

9.4 Conclusion

In this work, three different experimental techniques have been developed to study the vortex dynamics near the vortex-solid to vortex-liquid phase transition. An ac impedance measurement system was developed which can perform direct transport measurements on a superconducting sample to obtain the frequency dependent impedance over a frequency range of 10^2Hz to 10^7Hz with a sensitivity of 10nV. An ac magnetic susceptibility system was developed which can accurately measure the susceptibility of a superconducting sample over a frequency range of 1kHz to 2MHz. Third harmonic transmissivity measurements were applied to a sample with columnar defects, and by using a new analysis technique which identifies a characteristic time (τ) and crossover current (j), we are able to better understand the physical significance of the third harmonic data and for the first time apply critical scaling analysis to the third harmonic measurements to investigate the Bose-glass to vortex-liquid phase transition. The results derived from this analysis are shown to be consistent with previous measurements on a similar sample using the ac impedance techniques developed here. The establishment of these different measurement techniques has provided useful tools for future investigations of vortex dynamics in other superconducting systems.

References

[1] D. S. Fisher, M. P. Fisher, and D. A. Huse, Phys. Rev. B **43**, 130 (1991).

[2] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, F. Holtzberg, M. Konczykowski, C. C. Tsuei, and C. C. Chi, Physica A **200**, 374 (1993).

[3] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, D. A. Beam, and F. Holtzberg, Phys. Rev. B **49**, 4384 (1994).

[4] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6150 (1993); *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p. 413.

[5] D. S. Reed, N.-C. Yeh, M. Konczykowski, and A. V. Samoilov, submitted to Phys. Rev. Lett. (1994).

[6] D. R. Nelson and V. M. Vinokur, Phys. Rev. Lett. **68**, 2398 (1992); Phys. Rev. B **48**, 13060 (1993).

[7] D. S. Reed, N.-C. Yeh, W. Jiang, U. Kriplani, M. Konczykowski, and F. Holtzberg, submitted to Phys. Rev. Lett. (1994).

[8] N.-C. Yeh, W. Jiang, D. S. Reed, U. Kriplani, and F. Holtzberg, Phys. Rev. B **47**, 6146 (1993).

[9] W. Jiang, N.-C. Yeh, D. S. Reed, U. Kriplani, D. A. Beam, M. Konczykowski, T. A. Tombrello, and F. Holtzberg, Phys. Rev. Lett. **72,** 550 (1994).

Appendix A : Data Acquisition Program

A.1 Mscan main routine

The first program listing in this appendix is the main scan routine which is responsible for performing the measurement scan and ensuring that the correct data is taken at the correct time. This is the highest program level discussed in Chapter 4.8.2 and is responsible for setting all of the sample conditions (temperature, magnetic field, applied ac current, etc.), waiting for the sample temperature to become stable, and then performing the measurements. The scan routine is designed to be flexible enough to perform both dc and ac measurements, frequency dependent experiments, current dependent experiments, temperature dependent experiments, and other experiments which systematically scan one of the experimental variables. In order to allow such flexibility, the main routine consists of two primary sections: a menu section which allows the user to set up the desired scan, and the scan section which actually carries out the measurements.

```
                Scanning Routine

Scan Type: y04zfa01: Amplt   vs j_Freq -S
Name     : \data\ybco04\z_vs_f\y04zfa01.DAT   to model70 NONE
Magnet   : Superconducting

                Start        End      Increment
Amplt  :            0          0      ----          b     Y0
j_Freq :          100     2E+006      1.025068  x c
b_swtch:            1          0            -1    a
Temp   :           70        100          0.05    a     Y2
X_phase:            0          0      ----          b     Y1

Hold times            : Amplt  =0.100000 sec
Udder options
Delay between Readings: 0.010000 sec -- DELAY TIME TOO SMALL
Elephant Channels:
Wait (keep LN2 filled)
Quit
Kontinue Scan
Go to Scan
 Zero field at end
                  Approximate scan time:   21.36 min
(press a letter)
```

Figure A.1. Sample of the main menu listing of Mscan

Before proceeding with a description of the various menu options, it is helpful to define some notation used in the program. The "parameters" are any value which can be read from one of the meters such as sample impedance, temperature, magnetic field, or the frequency of the applied sample current. The "x-parameters" or "looping values" are those parameters which are set by the computer, and the "y-parameters" are values read by the computer and stored in a data file at every value of the "x-parameters." Figure A.1 shows a sample of the main menu which appears in Mscan that allows the desired scan to be set up. Pressing the first letter of each line allows access to changing the parameters listed in the line. For example, pressing "S" accesses "scan type" which allows the user to redefine which x-parameters and y-parameters are to be used. The choice of these parameters will be reflected in the parameter list which in the above example is amplitude, frequency, the transfer switch, temperature, and phase. In the above example, the "looping parameters" are frequency, transfer switch, and temperature, in that order. Therefore, the frequency would be incremented from its starting value of 100Hz to its ending value of 2MHz using a multiplicative step (note the "x" in the frequency line) of 1.02507. This multiplicative step corresponds to 401 points evenly spaced on a logarithmic frequency scale. At each of these frequency steps, the three y parameters (amplitude, phase, and temperature) as well as the frequency will be stored in the data output file. When the frequency reaches its end value (2MHz), the next looping parameter will be incremented (the switch) and the frequency scan will be repeated. This nested loop structure for the x-parameters is the reason for the term "looping parameters." In the above example, after frequency scans are completed with the transfer switch on and off, the temperature will be incremented by 0.03K and the scans will repeat. This series of scans would generate a series of isotherms similar to those shown in Figure 5.6 and Figure 8.2.

The other menu options shown in Figure A.1 allow various details of the scan to be set. The "Hold times" options sets a short delay in the scan which permits settings to stabilize. For example, after the magnetic field is set, we usually wait ~5min to ensure that the temperature fluctuations caused by induced currents in the smile probe have time to stabilize before any measurements are taken. The "Udder options" (other options) sets various meter options such as the integration time on the HP4194A which do not relate directly to the scan procedure. The "delay between readings" permits data to be taken at regular time intervals, such as reading the sample resistance once every 10 minutes while the system is cooling from room temperature. In most cases, we want to take data as fast as the measurements can be made reliably so this option is disabled by setting the delay to a value less than the actual time required to measure the data. The "Elephant channel" is not used in ac measurements, but is used in dc measurements to repeat voltage measurements at each setting of y-parameters. This repeated measurement helps to lower the noise level through signal averaging. The term "Elephant channel" is a joke which refers to one graduate student's fondness of elephants. The remaining options are relatively self explanatory: "go to scan" begins the measurement process, "quit" exits the program, "zero field at end" is set to yes or no to determine whether the magnetic field will be turned off after the scan is done. The following code displays the main menu, allows it to be changed, and then executes the scan procedure which has been set.

Listing of File "Mscan8h.c"

```
#include      "C:\lw\include\lwsystem.h"
#include      "C:\lw\include\formatio.h"
#include      "C:\lw\include\graphics.h"
#include      "C:\lw\include\mag_def.h"
#include      "C:\lw\include\temp_def.h"
#include      "C:\lw\include\scn_def.h"
#include      "C:\lw\include\lia_def.h"
#include      "C:\lw\include\optionh2.h"
#include <bios.h>
#include <string.h>
#include <stdio.h>
```

```
#define CURRENT_CHANNEL_OFFSET 10
#define HP_ADDRESS 8
#define LIA_ADDRESS 4
#define NVMETER_ADDRESS 6
#define CURRENT_ADDRESS 12
#define SCNNER_ADDRESS 18
#define MAGNET_ADDRESS 16
#define TSENSE_ADDRESS 2
#define CURR_CHAN 4
#define CURRENT_CHANNEL_OFFSET 10
#define TIME_CORRECTION .050
#define TRUE !0
#define NEXTSCAN 11
#define SCAN_DATA 12
#define TOTALPAR      10
#define FIRST_Y_PAR   5
#define PAUSE .50
#define PAUS_X_2 2
#define MAXPLOTS  100
#define MAXPOINTS 10000
#define MAX_CHANNEL 9
#define MIN_CHANNEL 0
#define LIA_TIMEOUT 120
#define MAG_RAMP_TIME 10
#define WAIT_FOR_TEMP 300
#define TEMP_ERROR .021
#define TEMP_STABLE_TIME 10
#define MAX_RANGE 3
#define MAGNET_RESOLUTION .001
#define MAGNET_DIG_OUT 0
#define FIELD_RATIO 1.1859
#define MAX_230CURRENT .1

/*define parameter values */
#define TEMPERATURE  1
#define FIELD_P 2
#define ITIME TOTALPAR-1
#define OSC_LEVEL 3
#define OSC_FREQUENCY 4
#define CURRENT 0
#define HP_SWITCH 5
#define HP_REAL 6
#define HP_IMAGINARY 7
#define ANGLE 8
#define FILL_INTERVAL 3600

/* keyboard definitions */
#define ESCAPE 0x11B
#define ENTER 0x1C0D

int setpar(); int getpar(); int abs(); int datsave(); double fabs(); long s_time();
double sqrt(); int reset_stack(); int push(); unsigned pull(); int CURRENT_init();
int CURRENT_set_scan();  int CURRENT_clear_srq();
int CURRENT_start_scan();  int CURRENT_fast_dwell();
```

```
int CURRENT_dwell(); int CURRENT_end_scan();
int CURRENT_set_cur_scan(); int CURRENT_close();
double CURRENT_amps_scan(); int NVMETER_init();
double NVMETER_fast_read(); double NVMETER_volts();
int NVMETER_status(); int NVMETER_set_par();
int NVMETER_close(); int SCNNER_init();
int SCNNER_close(); int SCNNER_reset();
int SCNNER_channel(); int MAGNET_init();
int MAGNET_close(); int MAGNET_ramp_up();
int MAGNET_ramping(); int MAGNET_heater();
double MAGNET_field(); int LIA_init(); int LIA_close();
double LIA_signal(); double LIA_oscillator(); int LIA_ready();
int LIA_auto(); FILE *Init_file(); int HP_init(); int HP_close();
double HP_read(); int HP_sweep(int type,float start,float end,int nop,int mult,double delay);
int TEMP_setpoint (float setpoint); int floppy(char *direct,char *nfame,float swtch);
unsigned wait_plus(long ret_time,int key); int repeat[10]; int temp_stable_time;
double temp_error; float tdata[30]; float sqdata[30]; float data[30]; int options[MAX_OPTIONS];
/* settings for high resolution temp control */
float tc_time,tc_gain,tc_reset,tc_max_wait,tc_stable,tc_error,tc_max_power,tc_rate;
double temp_min[3]; double temp_max[3]; long temp_num[3]; double temp_sum[3];
double temp_sqr[3];

void main()
{
        long start_scan;  /* keeps track of start of individual scan */
        int zero_field;  /* 0=manually,1=only at end, 2=whenever temp is increased */
        unsigned long temp_delset;  double hold[TOTALPAR+2];
        int cur_plot;  FILE *output;  char fpath[100],ppath[100];
        int continued; int low_field; int redraw; int col; int x_col; int y_par[TOTALPAR+2];
                /* marks which values were changed so only wait for those */
        int was_changed[TOTALPAR+2];
        double lim[TOTALPAR+2+CURR_CHAN][2]; double seg_lim[1+CURR_CHAN][2];
        int seg_step; int nstep[TOTALPAR+2]; int mult[TOTALPAR+2+CURR_CHAN];
        int aut[TOTALPAR+2]; char nfame[40]; char title[80]; unsigned key;
        int n; int port; int chan[TOTALPAR+2][10]; int totchan[TOTALPAR+2];
        int t; char m0[90]; char *testchar; int count; int ch; double start_cur;
        double volt; double tvolt[2]; double delset; double deltime; int aa,ab,ac;
        int averaged[30]; int l; int j; int loop; int a; int y__ch; FILE *par_file;
        int choice; double cur[2]; int b; int c; int d;int m; int p1[TOTALPAR+2];
        int p[TOTALPAR+2]; int order[TOTALPAR+2];
        float par[TOTALPAR+2+CURR_CHAN]; char direct[81]; char model70_dir[81];
        char *nstring; char *pname[TOTALPAR+2]; int cur_seg; int seg_max;

        seg_max = 0;
        nstring = "CTFJOAXBYIVR";
        /*pname relates the parameter number to the parameter name*/
        pname[CURRENT] = "Current";
        pname[TEMPERATURE] = "Temp   ";
        pname[FIELD_P] = "Field  ";
        pname[OSC_LEVEL] = "J_OscLv";
        pname[OSC_FREQUENCY] = "Osc_fre";
        pname[HP_SWITCH] = "A_swtch";  /*load voltage*/
        pname[HP_REAL] = "X_real ";  /* sample voltage/load voltage */
```

```
pname[HP_IMAGINARY] = "B_imag "; /* sample phase-load phase*/
pname[ANGLE] = "Y_angle";
pname[TOTALPAR-1] = "Itime  ";
pname[TOTALPAR] = "Voltage";
pname[TOTALPAR+1] = "Resistance";
/*the options[] control minor options in the scan routine and are set in the "Udder options"
routine in the main menu  this section sets the default settings*/
options[NV_OPTS] = -10;
options[NV_FILTER] = 0;
options[NV_DAMPING] = 4;
options[NV_WAIT] = 120;
options[NV_STABLIZE] = 4;
options[NV_ERROR] = 5;
options[TC_OPTS] = -10;
options[TC_WAIT] = 300;
options[TC_STABLIZE] = 15;
options[TC_LEAD_ERROR] = 2;
options[TC_TRAIL_ERROR] = 2;
options[TC_SAMPLE_CH] = 1;
options[CS_OPTS] = -10;
options[CS_VLIMIT] = 3;
options[LIA_OPTS] = -10;
options[LIA_EXPAND] = 0;
options[LIA_FILTER] = 0;
options[LIA_RESISTOR] = 0;
options[LIA_TIMECON] = 7;
options[HP_OPTS] = -10;
options[HP_INTEGRATE] = 1;
options[HP_TIMEOUT] = 120;
options[HP_AVERAGE] = 1;
/*set default settings for all scan parameters*/
strcpy(direct,"C:\\DATA");
strcpy(model70_dir,"C:\\easyplo");
strcpy(nfame,"SCAN00XX");
cur_plot = 0;
y_par[0] = 0;
continued = 0;
for (n = 0; n < 10; n++)
  repeat[n] = 1;
for (n = 0; n < CURR_CHAN; n++) {
  lim[n][0] = lim[n][1] = 0;
  mult[n] = 1;
} /** for n < CURR_CHAN .. **/
for (n = 0; n <= TOTALPAR+1; n++) {
  if ( n>0 )
    y_par[n] = -1;
  aut[n] = 0;
  hold[n] = 0;
  order[n] = n;
  lim[n+CURR_CHAN][0] = lim[n+CURR_CHAN][1] = 0;
  nstep[n] = 0;
  totchan[n] = 0;
  for (t = 0; t < 10; t++)
```

```
    chan[n][t] = 0;
  mult[n+CURR_CHAN] = 1;
} /** for n <= TOTALPAR+1 ..... **/
reset_stack();

/* use of aut:                                              */
/*     aut=0  => just read values, no setting, no waiting   */
/*     aut=1  => wait for values & read, no setting         */
/*     aut=2  => set values, and read without waiting       */
/*     aut=3 => set values, wait for them, read them        */

/* if "scan.par" file exists, read in all of the previous scan settings. The scan.par file must
be located in the current directory. This file is updated on quitting the program and
starting a scan. If this file does not exist, the default settings above are used*/
if ((par_file = fopen ("scan.par", "r+b")) != NULL) {
 fread (nfame,1,8,par_file);
 fread ((char *)nstep,sizeof(int),TOTALPAR+2,par_file);
 fread ((char *)lim,sizeof(double),2*(TOTALPAR+2+CURR_CHAN),par_file);
 fread ((char *)order,sizeof(int),TOTALPAR+2,par_file);
 fread ((char *)hold,sizeof(double),TOTALPAR+2,par_file);
 fread ((char *)chan,sizeof(int),10*(TOTALPAR+2),par_file);
 fread ((char *)totchan,sizeof(int),TOTALPAR+2,par_file);
 fread ((char *)&deltime,sizeof(double),1,par_file);
 fread ((char *)aut,sizeof(int),TOTALPAR+2,par_file);
 fread ((char *)y_par,sizeof(int),TOTALPAR+2,par_file);
 fread ((char *)&low_field,sizeof(int),1,par_file);
 fread ((char *)mult,sizeof(int),TOTALPAR+2+CURR_CHAN,par_file);
 fread ((char *)repeat,sizeof(int),10,par_file);
 fread ((char *)options,sizeof(int),MAX_OPTIONS,par_file);
 fread (direct,1,80,par_file);
 fread (model70_dir,1,80,par_file);
 fread ((char *)&zero_field,sizeof(int),1,par_file);
 fread ((char *)&tc_gain,sizeof(float),1,par_file);
 fread ((char *)&tc_reset,sizeof(float),1,par_file);
 fread ((char *)&tc_max_wait,sizeof(float),1,par_file);
 fread ((char *)&tc_stable,sizeof(float),1,par_file);
 fread ((char *)&tc_error,sizeof(float),1,par_file);
 fread ((char *)&tc_rate,sizeof(float),1,par_file);
 fread ((char *)&tc_max_power,sizeof(float),1,par_file);
 fread ((char *)&tc_time,sizeof(float),1,par_file);
 fclose (par_file);
} /** if par_file = fopen ... **/

if (low_field != 0)
  low_field = 1;
choice = 0;

while (choice != 'Q') {
  SetDisplayMode (0);
  choice = 0;
  for (n = 0; n <= TOTALPAR+1; n++)
    aut[n]=abs(aut[n]);
  while (choice != 'Q' && choice != 'G') {
```

```
cls ();
/* prevent stupid scan set-ups    */
hold[TOTALPAR] = hold[TOTALPAR+1];  /* nvmeter */
hold[3] = hold[4];  /* oscillator */
for (m = 0; m <= CURR_CHAN*2+1; m++) {
  if (lim[m/2][m%2] > MAX_230CURRENT)
    lim[m/2][m%2] = MAX_230CURRENT;
  if (lim[m/2][m%2] < -MAX_230CURRENT)
    lim[m/2][m%2] = -MAX_230CURRENT;
} /** for m <= CURR_CHAN*2=1 ... **/
/* print scan configuration */
sprintf (title,"%.8s_________",nfame);
if (low_field == 1)
  *m0 = 'L';
else
  *m0 = 'S';
sprintf (title, "%.8s: %.10s vs %.7s-%c", title,pname[y_par[0]],pname[order[0]],*m0);
printf ("\t\t\t Scanning Routine\n");
printf ("\n\tScan Type: %s\n", title);
printf ("\tName    : %s\\%s.DAT   to model70 %s\n", direct,nfame,model70_dir);
printf ("\tMagnet  : ");
if (low_field == 1)
  printf("Low Field");
else
  printf("Superconducting");
printf ("\n\n\t\t\tStart\tEnd\tIncrement");
for (n = -1; n < TOTALPAR+2; n++) {
  l = (n==-1) ? y_par[0]:order[n];
  for (a = 0; y_par[a] >= 0 && y_par[a] != l; a++);
  if (y_par[a] != l)
    a = -1;
  else if (aut[l] > 3)
    aut[l] = 0;
  if (nstep[l] != 0 || (aut[l] <= 3 && (a != 0 || n == -1))) {
    for (t = 0; t <= totchan[l]; t++) {
      ch = l + ((l == 0) ? chan[0][t]:CURR_CHAN);
      format(m0,lim[ch][0],lim[ch][1],nstep[l],mult[ch],aut[l]);
      if (l == 0)
        printf("\n\tCurren%i: %s",ch,m0);
      else
        printf("\n\t%7.7s: %s",pname[l],m0);
      if (a >= 0) printf("\tY%i",a);
      if (l != 0)
        t = 10;
      else if (seg_max > 0) {
        format(m0,seg_lim[ch][0],seg_lim[ch][1],seg_step,mult[ch],aut[l]);
        printf("\n\t      : %s",m0);
      }  /** else if seg-max > 0 **/
    } /** for t <= totchan[l ] ... **/
  } /** if (nstep[l] != 0 .... **/
} /** for (n = -1; n < TOTALPAR+2 ... **/
printf ("\n\n\tHold times          : %s=%f sec\n",pname[y_par[0]],hold[y_par[0]]);
printf ("\tUdder options\n");
```

```
    printf ("\tDelay between Readings: %f sec",deltime);
    delset = abs(repeat[y_par[0]]) * (hold[y_par[0]] + hold[order[0]] + PAUSE) *
             ( (y_par[0] >= TOTALPAR) ? 2:1) * (totchan[y_par[0]]+1);
    if (delset > deltime)
     printf(" -- DELAY TIME TOO SMALL");
    else
     delset = deltime;
    printf ("\n\tElephant Channels:");
    for (m = 0; m < 10; m++)
     if (repeat[m] != 1)
      printf("%i=%i: ",m,repeat[m]);
     printf("\n\tWait (keep LN2 filled)");
     printf ("\n\tQuit\n\tKontinue Scan\n\tGo to Scan");
     for (m = 0; m < TOTALPAR; m++)  /*calculate scan time*/
      delset *= (abs(nstep[m])+1);
    delset /= 60;
    choice = 0;

/* get menu option key.  after key is pressed, go to appropriate command option*/
    while ( !choice ) {
       choice = getkey ();
       if (choice > 90)
        choice -= 32;
       switch (choice) {
          case 'C':
            HP_conv(direct,nfame);
            break;
          case 'H':
             Set_holds(hold,pname,nstring);
             break;
          case 'L':
             if ( floppy(direct,nfame,0) >= 0 && floppy(direct,nfame,1) >= 0) {
              printf("\n\n\tfile copied to a:");
              wait_plus(s_time()+2000,0);
             }
             else {
               printf("\n\n\tERROR: cannot open files    (press any key)");
               _bios_keybrd(_KEYBRD_READ);
             }
             break;
          case 'W':
             printf("\n\nPress Esc to abort");
             init_time();                  /* must initialize timer for wait */
             MAGNET_init(16+33*low_field,-1.0);
             LN2_reset('n');
             m = 0;
             while (m == 0) {
                LN2_fill();
                if (wait_plus(s_time()+5000,2) == ESCAPE)
                  m = 1;
             }
             break;
          case 'D':
```

```
    printf ("\n0 = No delay, or\nNew delay time (sec):  ");
    ScanIn ("%f", &deltime);
    if (deltime <= 0)
      deltime = 0;
    else
      nstep[TOTALPAR-1] = fabs((lim[TOTALPAR-1+CURR_CHAN][1] -
        lim[TOTALPAR-1+CURR_CHAN][0]) / deltime);
    break;
  case 'U':
    Set_udder(options);
    break;
  case 'E':
    do {
     cls();
     printf("Repeat/reverse Channels:\n");
     for (m = 0; m < 10; m++)
       printf("%i:  %i\n",m,repeat[m]);
       printf("\n\n\t press key for channel (0-9)");
       if ((m = getkey() - '0') >= 0 && m <= 9) {
         printf("\n Enter repeats on channel %i (use - for reverse): ",m);
         ScanIn ("%i", &repeat[m]);
       }
    } while(m>=0 && m<=9);
    break;
  case 'G':
    continued = 0;
    break;
  case 'K':
    if (continued == 0) {
     printf("\n CANNOT CONTINUE");
     getkey();
    }
    else
     choice='G';
    break;
  case 'M':
    if (continued != 0) {
     printf("If used, cannot continue.\nPress Y to proceed anyway");
     m = getkey();
     if (m == 'Y' || m == 'y') {
       continued = 0;
     }
    }
    if (continued == 0)
      MAGNET_control(&low_field, &zero_field);
    break;
  case 'P':
    SetDisplayMode (1);
    PrintScreen ();
    break;
  case 'N':
    if (continued == 1) {
     printf("If used, cannot continue.\nPress Y to proceed anyway");
     m = getkey();
```

```
      if (m == 'Y' || m == 'y')
        continued=2;
    }
    if (continued != 1) {
     printf ("\nEnter Directory (or press enter for no change)\n\t");
     gets(m0);
     if (*m0 != 0)
      strncpy(direct,m0,75);
     printf ("\nEnter Name (8 characters):  ");
     gets(m0);
     if ((testchar = strchr(m0,'.')) != NULL)
      *testchar = 0;
     if (*m0 != 0)
      strncpy(nfame,m0,8);
     printf ("\nEnter Model 70 transfer Directory\n\t");
     gets(m0);
     if (*m0 != 0)
      strncpy(model70_dir,m0,75);
    }
    break;
  case 'S':
    if (continued != 0) {
     printf("If used, cannot continue.\nPress Y to proceed anyway");
     m = getkey();
     if (m == 'Y' || m == 'y')
       continued = 0;
    }
    if (continued == 0)
      Scan_set_up(order,y_par,aut,nstep,pname,nstring);
    break;
  case 'Q':
    break;
  default:
    n = strchr(nstring,choice)-nstring;
    if (n < 0 || n > TOTALPAR+1)
     choice = 0;
    if (choice != 0) {
      Set_scan(n,chan[n],totchan+n,nstep+n,mult+n,aut+n,pname[n],lim,0);
      if (n == 0) {
       printf("\n\nNumber of segments (1-2): ");
       seg_max = getche() -'0';
       seg_max = (seg_max>1) ? 1:0;
       if (seg_max > 0) {
         for (m = 0; m <= CURR_CHAN; m++)
            seg_lim[m][0] = lim[m][1];
         swap((char *)lim,(char*)seg_lim,(1+CURR_CHAN)*16);
         swap((char *)nstep,(char*)&seg_step,2);
          Set_scan(n,chan[n],totchan+n,nstep+n,mult+n,aut+n,pname[n],lim,1);
          swap((char *)lim,(char*)seg_lim,(1+CURR_CHAN)*16);
          swap((char *)nstep,(char*)&seg_step,2);
       }   /** if seg_max ... **/
      } /** if n == 0 ..**/
      if (n >= TOTALPAR) {
```

```
                totchan[0] = totchan[n];
                for (m = 0; m <= totchan[0]; m++)
                  chan[0][m] = chan[n][m];
              }
            } /** if choice != 0 **/
          } /** switch **/
        } /** while choice **/
      } /** while (choice != 'Q' && choice != 'G') **/
/*save scan settings to file "scan.par"*/
      par_file = fopen ("scan.par", "w+b");
      fwrite (nfame,1,8,par_file);
      fwrite ((char *)nstep,sizeof(int),TOTALPAR+2,par_file);
      fwrite ((char *)lim,sizeof(double),2*(TOTALPAR+2+CURR_CHAN),par_file);
      fwrite ((char *)order,sizeof(int),TOTALPAR+2,par_file);
      fwrite ((char *)hold,sizeof(double),TOTALPAR+2,par_file);
      fwrite ((char *)chan,sizeof(int),10*(TOTALPAR+2),par_file);
      fwrite ((char *)totchan,sizeof(int),TOTALPAR+2,par_file);
      fwrite ((char *)&deltime,sizeof(double),1,par_file);
      fwrite ((char *)aut,sizeof(int),TOTALPAR+2,par_file);
      fwrite ((char *)y_par,sizeof(int),TOTALPAR+2,par_file);
      fwrite ((char *)&low_field,sizeof(int),1,par_file);
      fwrite ((char *)mult,sizeof(int),TOTALPAR+2+CURR_CHAN,par_file);
      fwrite ((char *)repeat,sizeof(int),10,par_file);
      fwrite ((char *)options,sizeof(int),MAX_OPTIONS,par_file);
      fwrite (direct,1,80,par_file);
      fwrite (model70_dir,1,80,par_file);
      fwrite ((char *)&zero_field,sizeof(int),1,par_file);
      fwrite ((char *)&tc_gain,sizeof(float),1,par_file);
      fwrite ((char *)&tc_reset,sizeof(float),1,par_file);
      fwrite ((char *)&tc_max_wait,sizeof(float),1,par_file);
      fwrite ((char *)&tc_stable,sizeof(float),1,par_file);
      fwrite ((char *)&tc_error,sizeof(float),1,par_file);
      fwrite ((char *)&tc_rate,sizeof(float),1,par_file);
      fwrite ((char *)&tc_max_power,sizeof(float),1,par_file);
      fwrite ((char *)&tc_time,sizeof(float),1,par_file);
      fclose (par_file);
      init_time();
      delset=0.0;
      if (choice == 'G') {
        /*do some preliminary initializations for file formating, etc. */
        /*calculate number of columns being used */
        cur_seg = 0;
        if (order[0] == TOTALPAR-1 || order[0] == y_par[0])
          x_col = 1;
        else
          x_col = 2+totchan[order[0]];
        col = x_col;
        for (n = 0; y_par[n] >= 0; n++)
          col += totchan[y_par[n]]+1;
        /* initialize devices that are being used */
        printf("initialize LN2");
        MAGNET_init(16+33*low_field,-1.0);
        printf("--");
```

```
  LN2_reset('y');
  printf("-done\n");
  for (n = 0; n <= TOTALPAR+1; n++)  {
    if (aut[n] <= 3)
      p[n] = TRUE;
    else
      p[n] =! TRUE;
  } /** for n ...**/
  p[TOTALPAR-1] =! TRUE;   /*time does not get initialized */
  p[y_par[0]]=TRUE;
  if (p[y_par[0]] >= TOTALPAR && (chan[TOTALPAR][9] > 0 ||
            chan[TOTALPAR+1][9] > 0))
   p[0] = TRUE;
  printf("ready to initialize devices\n");
  n = Initialize(p,low_field,pname,chan,hold);
  printf("initialization done\n");
} /** if (choice == 'G')  **/

if (choice == 'G' && n >= 0)  {
 loop = 1;
 /* calculate initial values for all parameters (or reset for interrupted scan) */
 printf("\t\tInitializing Devices\n");
 /*set up HP sweep */
 HP_sweep(order[0],lim[order[0]+CURR_CHAN][0],lim[order[0]+CURR_CHAN][1],
          nstep[order[0]],mult[order[0]+CURR_CHAN],hold[order[0]]);
 for (n = TOTALPAR+1; n >= 0; n--)  {
   was_changed[n] =! TRUE;
   for (t = 0; t <= totchan[n]; t++) {
     if (n != 0)
       t = 10;
     ch = ( (n == 0) ? chan[0][t] : (n+CURR_CHAN));
     /* if continuing a previous scan */
     if (continued == 1 && nstep[n] != 0 && (mult[ch] != 0||lim[ch][0] != 0)) {
       if (nstep[n] > 0)
         b = nstep[n];
       else
         b = -.5*nstep[n];
       if (mult[ch] != 0)
         a = (par[ch]-lim[ch][0]) * b / (lim[ch][1]-lim[ch][0]);
       else
         a = log(par[ch]/lim[ch][0]) * b / log(lim[ch][1]/lim[ch][0]);
       p[n] = (p1[n] < 0 && nstep[n] < 0) ? 2*b+1-a : a;
     } /** if continuing ..**/
     else { /* for normal start of scan */
       p[n] = 0;
       par[ch] = lim[ch][0];
     }
   } /** for t <= totchan[n] .. **/
 } /** for (n = TOTALPAR+1; n >= 0; n--) **/

/*************************************************************/
/* Initialize data output file and store parameters at beginning of file  */
/*************************************************************/
```

```
 /* continued=1  => escape                                          */
 /* continued=2  => separate file,same screen                       */
 /* continued=0  => new scan                                        */
 if (loop > -1)
  if ((output = Init_file(direct,nfame,title,pname,y_par,aut,order,continued,par,chan,
                          totchan,hold,repeat)) == NULL)
    loop = -1;
 /***set initial values, but do NOT set field yet ****/
 volt = 0.0;  /*used to keep track of longest hold time needed */
 for (n = TOTALPAR+1; n >=0; n--) {
   if (aut[n] >= 2 && aut[n] <= 3 && (n != FIELD_P || zero_field != 2)) {
     printf("\n\t\t  Device %s=%g\n",pname[0],par[n+((n==0)?0:CURR_CHAN)]);
     if (volt < hold[n] && n != 3 && n != 4)
       volt = hold[n];
     if (zero_field == 2 && n == TEMPERATURE && order[0] != TEMPERATURE)
       Set_zero_field(low_field,0);  /* zero field before changing temperature */
     for (m = 0; setpar(n,&par[n+((n == 0) ? 0 : CURR_CHAN)],
              chan[n][0],chan[n][totchan[n]],order[0]) < 0 && m < 5;m++) {
       printf("%i",m+1);
       GEN_init(0);
       wait_plus(s_time()+1000,0);
     } /** for m .. **/
     was_changed[n] = TRUE;
     if (m >= 5) {
       printf("\n\tCannot set device %s\n",pname[n]);
       printf("\t\t Press C to continue, any other to abort scan\n");
       m = getkey();
       if (m != 'c' && m != 'C')
         loop = -1;
     } /** if  m >= 5 **/
   } /** if aut .. **/
 } /** for  n = TOTALPAR+1 ..**/
 init_time();
 delset = 0.0;
 if (continued == 0)   /* if scan is not being continued*/
   count = 0;
 printf("Is LN2 filled? (Y/N) ");
 while ((n = toupper(getchar())) != 'Y' && n != 'N');
 LN2_reset(n);
 /*************************************************************/
 /* Start initial scan loop to loop through all possible X parameters  */
 /*************************************************************/
 while (loop > -1) {
     loop = 1;
     for (n = 1; n < TOTALPAR && loop > -1; n++)
       if ((order[n] != FIELD_P || zero_field != 2)  && abs(aut[order[n]]) <= 3 &&
           order[n] < TOTALPAR-5) {  /* set field only after everything is stable */
         m = -1;
         a = 0;
         loop = 1;
         /***  Make sure all parameters are at the correct value  ***/
         if (was_changed[order[n]]) {
           while (a < 5 && loop == 1 && (m = getpar(order[n],&par[order[n] +
```

```
            ((order[n] == 0) ? 0 : CURR_CHAN)] ,aut[order[n]], order[0],
            chan[order[n]][0], par, hold[order[n]])) != 0) {
        if (m < 0)  {
          a++;
          GEN_init(0);
        }
        else if (m == 0x11B)
          loop = -1;
        else if (m == 0x1C0D)
          loop = 2;
      } /** while a < 5 ..**/
      was_changed[order[n]] =! TRUE;
    } /** if was_changed ..**/
    if (m < 0)
      printf("\n\tCannot get %s value\n",pname[order[n]]);
  } /** if ((order[n] != FIELD  ... **/

  if (loop > -1) {
    loop = 1;
    /****  Wait for hold times on all values changed  ****/
    /* first, set field */
    if (zero_field == 2)  {   /* if field was zeroed before changing temperature */
      n = FIELD_P;
      setpar(FIELD_P,&par[FIELD_P+CURR_CHAN],
            chan[FIELD_P][0],chan[FIELD_P][totchan[FIELD_P]],order[0]);
      getpar(n, &par[n+((n==0)?0:CURR_CHAN)],
            aut[n],order[0],chan[n][0],par,hold[n]);
      /**** need to make sure temp & field are stable ******/
      n = TEMPERATURE;
      if (abs(aut[n]) <= 3 && aut[n] < TOTALPAR-5) {
        m = -1;
        a = 0;
        loop = 1;
        /***** Make sure all parameters are at the correct value  *****/
        if (was_changed[n]) {
          while (a < 5 && loop == 1 &&
                (m = getpar(n, &par[n+((n==0)?0:CURR_CHAN)],
                 aut[n],order[0],chan[n][0],par,hold[n])) != 0) {
            if (m < 0) {
              a++;
              GEN_init(0);
            }
            else if (m == 0x11B)
              loop = -1;
            else if (m == 0x1C0D)
              loop = 2;
          } /** while a < 5 ...**/
          was_changed[n] =! TRUE;
        } /** if  was_changed ..**/
        if (m < 0)
          printf("\n\tCannot get %s value\n",pname[n]);
      } /** if abs .. **/
    } /** if  zero ... **/
```

```
    temp_delset = s_time();
    while (loop == 1 && (key = wait_plus(temp_delset+(long)1000*volt,1)) != 0) {
       if (key == 0x11B)
         loop=-1;
       else if(key==0x1C0D)
         loop=2;
    } /** while **/
    temp_delset = s_time();
    volt = 0;
 } /**  if (loop > -1) **/
 m = 0;
 /*********************************/
 /* Main scan loop for individual scan  */
 /*********************************/
 LN2_fill();  /* fill LN2 before beginning a scan */
SetTitle (title);
start_scan = s_time();
for (n = 0; n <= 2; n++)  {
   temp_min[n] = 1e5;
   temp_max[n] = -1e5;
   temp_num[n] = 0;
   temp_sum[n] = 0.0;
   temp_sqr[n] = 0.0;
}
HP_start_sweep();
while (loop > 0)  {
   if ((s_time()-start_scan) / 1000  > FILL_INTERVAL)
     LN2_fill();
   n = 0;
   a = -1;
   /****   Make sure x parameter is at value  *****/
   if (aut[order[0]] == 1 || aut[order[0]] == 3)  {
    while(a != 0x11B && n < 5 && a < 0)  {
       a = 0;
       for (m = 0; m <= totchan[order[0]] && a != 0x11B && a >= 0; m++) {
          if ((a = getpar(order[0],&par[order[0]+((order[0] == 0) ? 0:CURR_CHAN)],
              aut[order[0]],order[0],m,par,hold[order[0]])) < 0) {
            GEN_init(0);
            n++;
          }
          else if (a == 0x11B)  {
            loop = -1;
            for (m = 0; m <= TOTALPAR+1; m++) {
              if (p[m] < -.5*nstep[m])
                p1[m] = -1;
              else
                p1[m] = 0;
            } /** for m ...  **/
            continued = 1;
            n = 6;
          }  /** else **/
          else if (m == 0x1C0D)
           n = 6;
```

```
        } /** f or m ..**/
      }   /** while a .. **/
      if (a < 0)
        printf("Cannot get %s value\n",pname[order[0]]);
    }  /** if aut ...**/
    n = 0;
    /****  wait for delay & check keyboard while waiting  ****/
  while (loop > -1 && volt >= 0 && n == 0) {
    if (volt > 0) {
      temp_delset = s_time();
      tvolt[1] = volt;
    }
    else {
      temp_delset = delset*1000;
      tvolt[1] = deltime-TIME_CORRECTION;
      if (order[0] == ITIME && tvolt[1] < par[ITIME+CURR_CHAN] - delset)
        tvolt[1] = par[ITIME+CURR_CHAN] - delset;
    }
    l = 0;
    while (l == 0 && n == 0) {
      if ((key = pull()) == 0)
        key=wait_plus(temp_delset+(long)tvolt[1]*1000,-1);
      switch(key) {
        case 0:
          l = 1;
          break;
        case 0x1C0D:
          n = 1;
          break;
        case 0x11B :
          loop = -1;
          for (m = 0; m <= TOTALPAR+1; m++) {
            if (p[m] < -.5*nstep[m])
              p1[m] = -1;
            else
              p1[m] = 0;
          }
          n = continued = 1;
          break;
      }   /** switch **/
    }    /** while **/
    volt = (volt == 0) ? -1 : 0;
  }  /** while loop ... **/
delset = s_time() / 1000;
volt = 0;
if (loop > -1) {
   /***  read device & plot point ****/
  a = 0;
  b = 1;
  for (n = 1; n <= col+1; n++)
    data[n] = sqdata[n] = averaged[n] = 0;
  while (b <= abs(repeat[0]) && a != 0x11B)  {  /*global repeat (channel 0) */
         /* read all y parameters */
```

```
            for (n = m = 0; y_par[m] >= 0 && a != 0x11B; m++) {
              for (t = 0; t <= totchan[y_par[m]] && a != 0x11B; t++) { /* channels */
                aa = chan[y_par[m]][t];
                        /*channel repeat */
                for (ab = 1; ab <= abs(repeat[aa]) && a != 0x11B; ab++) {
                  if ((a = getpar(y_par[m],&tdata[x_col+n],0,order[0],aa,par,
                      (b == 1 || y_par[m] >= TOTALPAR) ? hold[y_par[m]] : 0)) < 0)
                  printf("Can not get %s value (c)\n",pname[y_par[m]]);
                  if (repeat[aa] < 0)
                    tdata[x_col+n] *= -1;
                  data[x_col+n] += tdata[x_col+n];
                  sqdata[x_col+n] += (tdata[x_col+n]*tdata[x_col+n]);
                  averaged[x_col+n]++;
                } /** for ab = ..* */
                n++;
              } /** for t = 0 ...**/
            } /** for n = m .. **/
            if (order[0] != TOTALPAR-1 && a != 0x11B && order[0] != y_par[0]) {
              m = order[0];
              for (n = 0; n <= totchan[order[0]]; n++) {
                if (abs(aut[order[0]]) <= 3) {
                  if ((a = getpar(order[0],&par[order[0]+((order[0] == 0) ? 0 : CURR_CHAN)]
                      ,0,m,chan[order[0]][n],par,0)) < 0 && a != -117)
                    printf("Can not get %s value\n",order[0]);
                }
                else
                  a = 0;
                tdata[n+1] = par[order[0]+((order[0] == 0) ? 0 : CURR_CHAN)];
                  /* sample phase-load phase*/
                if (order[0] == TOTALPAR-2 && averaged[1+n] >= 1)
                  tdata[1+n] = branch(tdata[1+n],data[1+n]/averaged[1+n],360.0);
                data[n+1] += tdata[n+1];
                sqdata[n+1] += tdata[n+1]*tdata[n+1];
                averaged[n+1]++;
              } /** for n = ... **/
            } /** if order .. **/
            b++;
          } /* (while) number of times to repeat measurement ('elephant channel')*/
          if (a == 0x11b) {
            loop = -1;
            for (m = 0; m <= TOTALPAR+1; m++) {
              if (p[m] < -.5*nstep[m])
                p1[m] = -1;
              else
                p1[m] = 0;
            }
            n = continued = 1;
          } /** if a == .. **/
        } /** if loop > -1 **/
        /**** calculate average and error ****/
        if (loop > -1) {
          if (count == 0)
            fprintf(output,"/* avg: 0");
```

```
for (n = 1; n < col; n++)  {
   if (count == 0)
     fprintf(output,",%i",averaged[n]);
    if (averaged[n] > 0)  {
     data[n] /= averaged[n];
     sqdata[n] /= averaged[n];
    }
    else
     data[n] = sqdata[n] = 0;
    if (sqdata[n] <= data[n] * data[n] || averaged[n] == 1)
      sqdata[n] = 0;
    else  {
      sqdata[n] = sqrt((double)sqdata[n] - data[n] * data[n]);
       if (data[n] != 0)
         sqdata[n] /= data[n];
    }
}  /** for n = 1 .. **/
if (count == 0)
   fprintf(output,"\n#");
*data = s_time() / 1000.0;
b = totchan[y_par[0]]+1;
/****   increment counters to next point ****/
for (m = 1; m < col; m++){
 if (averaged[m] <= 1) {
   if (fwrite((char *)(data+m),sizeof(float),1,output) < 1) {
     beep();
     printf("WARNING: points not saved!");
   }  /** if fwrite .. **/
 }  /** if average ... **/
 else {
   if (fwrite((char *)(data+m),sizeof(float),1,output) < 1
      || fwrite((char *)(sqdata+m),sizeof(float),1,output) < 1) {
     beep();
     printf("WARNING: points not saved!");
   }
 }  /** else **/
}  /** for m .. **/
count++;
loop  = -2;
for (n = 0; loop == -2; n++)  {
   if (n > TOTALPAR+1 ) {
     loop == -1;
     continued = 2;
   }
   else  {
     p[order[n]]++;
     if (order[n] == TEMPERATURE &&
        (par[TEMPERATURE+CURR_CHAN] < 95 ))  {
       if (par[TEMPERATURE+CURR_CHAN] < 88) {
         p[TEMPERATURE] += 4;
         printf("  skip 4 scans");
       }
       if (par[TEMPERATURE+CURR_CHAN] < 80)  {
```

```
            p[TEMPERATURE] += 15;
            printf("  skip 19 scans");
            HP_set("NOA=",(float)1);
          }
          printf("\n");
        }  /** if order .. **/

        if (p[order[n]] > abs (nstep[order[n]])) {
          if (order[n]==0 && seg_max > 0)  {
            swap((char *)lim,(char*)seg_lim,(1+CURR_CHAN)*16);
            swap((char *)nstep,(char*)&seg_step,2);
            cur_seg++;
          }
          p[order[n]] = 0;
          if (order[n] == 0 && cur_seg <= seg_max && seg_max > 0)
            loop = n+1;
          else {
            cur_seg = 0;
          }
        } /** if p[order] ... **/
        else
          loop=n+1;
      } /** else **/
    }  /** for n .. **/
    if (loop != 1) { /*end of single scan -- reset for new file*/
      rewind(output);
      for (m = 0; m < 3;m++)
        if (temp_num[m] > 0) {
          temp_sum[m] /= temp_num[m];
          fprintf(output,"/* T%i=%.4f+-%.4f; min=%.3f, max=%.3f, num=%i\r\n",m,
                  temp_sum[m],sqrt(fabs(temp_sqr[m]/temp_num[m]-temp_sum[m]
                          *temp_sum[m])),temp_min[m],temp_max[m],temp_num[m]);
        }
      fclose(output);
      if (floppy(direct,nfame,par[HP_SWITCH+CURR_CHAN]) < 0)  {
        printf("FILE %s NOT SAVED TO FLOPPY",nfame);
        beep();
      }
      else
        printf("File saved to floppy\n");
    }  /** if loop .. **/
    /****  calculate next scan value ****/
    for (m = 0; m <= loop-1; m++)  {
      for (t = 0; t <= totchan[0]; t++)  {
        if (order[m] != 0)
          t = 10;
        ch = ((order[m]==0) ? chan[0][t] : (order[m]+CURR_CHAN));
        if (nstep[order[m]] > 0)  {
          a = p[order[m]];
          b = nstep[order[m]];
        }
        else if (nstep[order[m]] == 0)  {
          a = 0;
```

```
        b = 1;
      }
      else {
        if (order[m] == TEMPERATURE && (abs(aut[order[m]]) == 1 ||
            abs(aut[TEMPERATURE] == 3))) {
          aut[order[m]]=abs(aut[order[m]]);
          if (p[order[m]] > -.5*nstep[order[m]])
            aut[order[m]] *= -1;
          if (lim[ch][0] > lim[ch][1])
            aut[order[m]] *= -1;
        }
        b = -.5*nstep[order[m]];
        if (p[order[m]] <= -.5*nstep[order[m]])
          a = p[order[m]];
        else
          a = -nstep[order[m]]-p[order[m]];
      } /** else **/
      if (mult[ch] == 0)
        par[ch] = lim[ch][0] * pow(lim[ch][1] / lim[ch][0], a/(double)b);
      else
        par[ch] = lim[ch][0] + (lim[ch][1] - lim[ch][0]) * a/(double)b;
    } /** for t .. **/
    /**** set new parameters *****/
    if ((abs(aut[order[m]]) == 2 || abs(aut[order[m]]) == 3)) {
      if (volt < hold[order[m]] && order[m] != 3 && order[m] != 4)
        volt = hold[order[m]];
      if (order[m] == TEMPERATURE && m != 0 && zero_field == 2)
        Set_zero_field(low_field,0);
      for (a = 0; setpar(order[m],par + (order[m]+((order[m]==0) ? 0 : CURR_CHAN)),
          chan[order[m]][0],chan[order[m]][totchan[order[m]]],order[0]) < 0 && a < 5;
          a++) {
        GEN_init(0);
        wait_plus(s_time()+1000,0);
      }
      was_changed[order[m]] = TRUE;
      if (a >= 5)
        printf("Cannot set device %s\n",pname[order[m]]);
    } /** if abs .. **/
  } /** for m .. **/
  if (loop > -1 && loop != 1) { /*end of single scan -- reset for new file*/
    /*************************************************************/
    /* Initialize data output file and store parameters at beginning of file */
    /*************************************************************/
    /* continued=1 => escape                                     */
    /* continued=2 => separate file,same screen                  */
    /* continued=0 => new scan                                   */
    if ((output = Init_file(direct,nfame,title,pname,y_par,aut,order,0,par,chan,
        totchan,hold,repeat)) == NULL)
      loop = -1;
    count = 0;
  } /** if loop ... **/

if (loop > 1)
```

```
                loop = 0;
              }     /******** if(loop>-1) : calculate values if scan not aborted */
            }     /******** while(loop>0) : loop for single scan */
            init_time();
            delset = 0.0;
          }     /******** while(loop>-1) : loops all x parameters */
          fclose(output);
          printf("closing current devices\n");
          CURRENT_close ();
          printf("close scnner\n");
          SCNNER_close ();
          printf("close voltmeter\n");
          NVMETER_close ();
          printf("Setting temperature to %f\n",lim[TEMPERATURE+CURR_CHAN][0]);
          printf("close temp control\n");
          TEMP_close ();
          printf("close tsense\n");
          TSENSE_close ();
          if (zero_field > 0) {
           printf("zero field\n");
           Set_zero_field(low_field,0);
          }
          printf("close magnet\n");
          MAGNET_close();
          printf("close HP\n");
          HP_close();
          if (LIA_accessed()) {
           printf("close LIA\n");
           LIA_close();
          }
          printf("\n\n\n\tSCAN IS DONE!!!!\n\n\tKeeping LN2 filled");
          printf("\n\nPress ESCAPE to abort");
          m = 0;
          while (m == 0)  {
            LN2_fill();
            if (wait_plus(s_time()+5000,2) == ESCAPE)
              m = 1;
          }
        }  /** if (choice == 'G' && n >= 0)  **/
    }  /** while (choice != 'Q') **/
}
/*****************************************/
/*format is a minor routine which is used to print    */
/*  the start, end, increment line for each parameter  */
/*  it is a separate routine because it is used both to    */
/*  display the main menu and for changing parameter*/
/*  values */
int format(m0,lim0,lim1,nstep,mult,aut)
char *m0;
double lim0,lim1;
int nstep,mult;
{
        int m;
```

```
        sprintf(m0,"%10.4G %10.4G\t",lim0,lim1);
        if (nstep < 0)
          m = abs(nstep) / 2;
        else
          m = nstep;
        /* note on mult:                                    */
        /* mult=1 -> linear steps, linear graph scale       */
        /* mult=0 -> log steps, log graph scale             */
        /* mult=2 -> linear steps, log graph scale          */
        /*         (mult=2 used if nstep==0)                */
        if (nstep == 0)
          sprintf(m0,"%s----      ",m0);
        else if (mult == 0)
          sprintf(m0,"%s %10.4Gx",m0,pow(lim1/lim0,(double)1/(double)m));
        else
          sprintf(m0,"%s %10.4G",m0,(lim1-lim0)/m);
        if (nstep < 0)
          sprintf(m0,"%sr",m0);
        if (mult == 2)
         sprintf(m0,"%sln",m0);
        sprintf(m0,"%s\t%x",m0,aut+10);
}
```

A.2 Getpar/Setpar Routines

The subroutines Getpar() and Setpar() are the first sublevel of routines in Mscan as described in Chapter 4.8.2. These routines establish a link between the primary Mscan routine in section A.1 and the device calls in section A.3. Setpar() allows Mscan to use a single routine to set any scan parameter by simply changing the number indicating which device is to be set. Similarly, Getpar() can read the value from any device or wait for any device to stabilize before taking the reading. The following routines are contained in the file mscnsub3.c and in addition to Setpar() and Getpar include several small routines used for reading the system clock and filling liquid nitrogen into the dewar.

```
#include     "C:\lw\include\lwsystem.h"
#include     "C:\lw\include\formatio.h"
#include     "C:\lw\include\graphics.h"
#include     "C:\lw\include\mag_def.h"
#include     "C:\lw\include\temp_def.h"
#include     "C:\lw\include\scn_def.h"
#include     "C:\lw\include\lia_def.h"
#include     "C:\lw\include\options.h"

#include <bios.h>
#include <string.h>
#include <stdio.h>
#include <sys\timeb.h>
#include <time.h>

#define HP_ADDRESS 8
#define MAX_FLOAT 3.4E38
#define LOAD_RESISTANCE 1000
#define LIA_ADDRESS 4
#define NVMETER_ADDRESS 6
#define CURRENT_ADDRESS 12
#define SCNNER_ADDRESS 18
#define MAGNET_ADDRESS 16
#define TSENSE_ADDRESS 2
#define CURRENT_CHANNEL_OFFSET 10
#define CURR_CHAN 4
#define TIME_CORRECTION .050
#define TRUE !0
#define FALSE 0
#define NEXTSCAN 11
#define SCAN_DATA 12
#define TOTALPAR      10
#define FIRST_Y_PAR   5
#define PAUSE .50
```

```
#define PAUS_X_2 2
#define MAXPLOTS  100
#define MAXPOINTS 10000
#define MAX_CHANNEL 9
#define MIN_CHANNEL 0
#define LIA_TIMEOUT 180
#define MAG_RAMP_TIME 10
#define TEMP_WAIT_INT 30
#define MAX_RANGE 3
#define MAGNET_RESOLUTION .001
#define MAGNET_DIG_OUT 0
#define FIELD_RATIO 1.1859
#define MAX_CURRENT 75.9

/*define parameter values */
#define TEMPERATURE  1
#define FIELD_P 2
#define ITIME TOTALPAR-1
#define OSC_LEVEL 3
#define OSC_FREQUENCY 4
#define CURRENT 0
#define HP_SWITCH 5
#define HP_REAL 6
#define HP_IMAGINARY 7
#define ANGLE 8

/*TEMP_read() channel codes*/
#define T_GAIN -5
#define T_RATE -4
#define T_RESET -3
#define T_POWER -1
#define T_SETPOINT -2
#define T_CONTROL 0
#define T_SAMPLE 1

/* keyboard codes */
#define ESCAPE 0x11B
#define ENTER 0x1C0D
#define FULL_SIGNAL 2.40  /*signal for LN2 FULL (used only if >=0) */
        * change in signal whel LN2 is full. NOTE: this is ignored if FULL_SIGNAL > 0  */
#define FULL_STEP .15
#define FILL_INTERVAL 6300  /*time (sec) between LN2 fills*/

int setpar(); int getpar(); int abs(); int datsave(); double fabs(); double branch();
long s_time();long f_time(); unsigned wait_plus(long ret_time,int key);
double TEMP_read(int channel); int TEMP_setpoint (float setpoint); double TEMP_get_high();
unsigned TEMP_at_setpoint (unsigned long,int,double,double,double,int);
unsigned TEMP_high_control (unsigned long time0,int timeout,double sp,double n_error,double
p_error,float stable);  int TEMP_high_setpoint(double sp);
int CURRENT_init(); int CURRENT_set_scan(); int CURRENT_clear_srq();
int CURRENT_start_scan(); int CURRENT_fast_dwell(); int CURRENT_dwell();
int CURRENT_end_scan(); int CURRENT_set_cur_scan(); int CURRENT_close();
double CURRENT_amps_scan();
```

```
double NVMETER_read2(); int NVMETER_init(); double NVMETER_fast_read();
double NVMETER_volts(); int NVMETER_status(); int NVMETER_set_par();
int NVMETER_close();double NVMETER_stable ();
int SCNNER_init(); int SCNNER_close(); int SCNNER_reset(); int SCNNER_channel();
int MAGNET_init(); int MAGNET_close(); int MAGNET_ramp_up(); int MAGNET_ramping();
int MAGNET_heater(); double MAGNET_field(); double MAGNET_read_aport();
double LIA_read(); int LIA_init(); int LIA_close(); double LIA_signal();
double LIA_oscillator(); int LIA_ready(); int LIA_auto();
float HP_getpar(); double HP_read();
extern int options[MAX_OPTIONS];
/* settings for high resolution temp control */
extern float tc_gain, tc_reset, tc_max_wait, tc_stable, tc_error;
```

int setpar (n, val, minchan, maxchan, x1_par)

```
int n;                  /* which parameter to set*/
int minchan,maxchan;    /* channel numbers to set*/
float *val;             /* value to set */
int x1_par;             /*first x parameter -- primary sweep variable*/
{
        int error; int l, m, j; double diff; int zero; char dummy[5];
        zero=0;
        error = 0;
        if (n == TEMPERATURE) {
          diff = .01 * (int)(*val * 100 + .5);
          error = TEMP_setpoint ((float)diff);
          if (options[TC_SAMPLE_CH] >= 0)
            TEMP_high_setpoint((double)*val+options[TC_LEAD_ERROR]*.001); /* include offset */
          if (error >= 0)
            wait_plus(s_time()+1000,0);
          if (error < 0 || fabs(TEMP_read(T_SETPOINT) - diff) >= .01)
            error = -1;
        }
        else if (n == ANGLE) {
          printf("set angle to: %f (press enter)",*val);
          gets(dummy);
          error = 0;
        }
        else if (n == OSC_LEVEL) {
          if (x1_par != OSC_LEVEL)
            HP_set("OSC=",*val);
        }
        else if (n == OSC_FREQUENCY) {
          /* set up oscillator level */
          if (x1_par == OSC_LEVEL)
            HP_set2("OSC=",*val,"V");
          else if (x1_par != OSC_FREQUENCY) {  /*set up to measure a single point*/
            HP_set("START=",*val);     /* sweep up:down*/
            HP_set("STOP=",*val);      /* sweep up:down*/
            HP_set("NOP=",(float)2);   /* number of points*/
          }
        }
        else if (n == HP_SWITCH)
          error = SWITCH_set(*val); /* set cable switch */
```

```
        else if (n == FIELD_P)
          error = MAGNET_ramp_up(1,*val);
        else if (n == CURRENT) {
          error = 0;
          for (m = minchan; m <= maxchan; m++) {
            for (l = 1; l <= 2; l++) {
              j = (l == 1) ? 1 : -1;
              if (CURRENT_set_cur_scan (l+3*m, val[m]*j) < 0)
                error = -1;
            }
          }
          for (m = minchan; m <= maxchan; m++) {
            j = 1;
            for (l = 1; l <= 2; l++) {
              j = (l == 1) ? 1 : -1;
              diff = fabs(val[m]*j-CURRENT_amps_scan(l+3*m));
              if (diff > 1e-12 && (val[m] == 0 || diff / fabs(val[m]) > .0025)) {
                error = -1;
                printf ("current channel %i not set\n",l+m*3);
              }
            }
          }
        }
        return error;
}
```

int getpar (l, val, aut, first, channel, current, hold)

```
double hold;
float *current;
int aut;
int first;
int l;
int channel;
float *val;
{
        /* use of aut:                                          */
        /*     aut=0  => just read values, no setting, no waiting   */
        /*     aut=1  => wait for values & read, no setting         */
        /*     aut=2  => set values, and read without waiting       */
        /*     aut=3  => set values, wait for them, read them       */
        /*     for aut=1, assume value to wait for is >*val         */
        /*     for aut=-1, assume value to wait for is <*val        */
        float val2; float p_error, n_error; float test;
        int m; int error; int a, b, c, j, n; int zero;
        unsigned um; double tvolt[2]; char m0[100];
        unsigned long time1; unsigned long time0;

        zero = 0;
        if (aut > 3  && l < TOTALPAR && l != 3)
          return 0;
        error = 0;
        test = * val;
        n = 0;
```

```
/*************************************************************/
/** Get Magnet Current                                      *********/
/*************************************************************/
switch(l) {
  case FIELD_P:
    if (aut == 1 || aut == 3) {
      for (m = 0; m <= MAG_RAMP_TIME / 10; m++)  {
        if (MAGNET_ramping(10) > 0)  {
          error = 0;
          m = MAG_RAMP_TIME;
        }
        else if ((a = _bios_keybrd(_KEYBRD_READY)) != 0) {
          if (a == 0x11B || a == 0x1C0D)
            return _bios_keybrd(_KEYBRD_READ);
          else
            push(_bios_keybrd(_KEYBRD_READ));
        }
      }
      if (aut == 3)  {
       if (MAGNET_heater(0) >= 0 && first != 2)
         MAGNET_ramp_zero(0);
       else
         wait_plus(s_time()+2000,0);
      }
    }
    error = 0;
    *val = MAGNET_field();
    if (*val < -1e9)
      error = -1;
    break;
/*************************************************************/
/***  Get Temperature                                    ********/
/*************************************************************/
  case TEMPERATURE:
    error = 0;
    if (abs(aut) == 1 || abs(aut) == 3)  {
      val2 = *val;
      if (abs(aut) == 3 || *val < 0)
        val2 = -500;
      if (options[TC_TRAIL_ERROR] < 0)  {
        p_error = 1e6;
        n_error = -1e6;
      }
      else  {
        p_error = (options[TC_LEAD_ERROR] + options[TC_TRAIL_ERROR]) * .001;
        n_error = (options[TC_LEAD_ERROR] - options[TC_TRAIL_ERROR]) * .001;
      }
      time1 = s_time();
      while ((um = TEMP_at_setpoint(time1, options[TC_WAIT], val2, n_error,
                    p_error,options[TC_STABLIZE])) > 0 && um < 0xFFFF)  {
        if (um == 0x11B)
          return um;
        else if (um == 0x1C0D)
          break;
```

```
        else
          push(um);
      }
      if (options[TC_SAMPLE_CH] >= 0)  { /* if using high res temperature control */
       if (tc_error < 0)  {
         p_error = 1e6;
         n_error = -1e6;
       }
       else{
         p_error = options[TC_LEAD_ERROR] * .001+ tc_error;
         n_error = options[TC_LEAD_ERROR] * .001 - tc_error;
       }
       time1 = s_time();
       if (options[LIA_RESISTOR] >= 0)  /* if lia auto measure is on */
         LIA_auto(LIA_SAMPLE+LIA_LOAD,MEAS);
       printf("waiting for T=%f+ %f - %f lia_accessed=%i\n",
              *val,p_error,n_error,LIA_accessed());
       while ((um = TEMP_high_control(time1,tc_max_wait,*val,n_error,
                    p_error,tc_stable)) > 0 && um < 0xFFFF)  {
         if (um == 0x11B)
           return um;
         else if (um == 0x1C0D)
           break;
       }
      }
    }
    if (options[TC_SAMPLE_CH] >= 0)   /* if using high res temperature control*/
      *val=TEMP_get_high();
    else
      *val=TEMP_get_temp(channel,&first);
    if (first < 0 || first == 117)
      error = first;
    else
      error=0;
    if (*val < 0)  {
      printf("got negative temp: %f",*val);
      error = -1;
    }
    break;
/*************************************************************/
/***   Get Current                                         ********/
/*************************************************************/
/* current source has only auto mode at present time */
case CURRENT:
    error = 0;
    if (channel >= 0)
      a = 3;
    else
      a = -3;
    *val = -1.0;
    for (m = 0; m < 5 && *val <= -1.0; m++)  {
      *val = CURRENT_amps_scan (a*channel+1);
      if (*val <= -1.0)
```

```
        printf("error reading _amps_scan(%i)",a*channel+1);
    }
    if (*val < -.99)
      error = -1;
    break;
/*************************************************************/
/***              Get HP/LIA info                  ********/
/*************************************************************/
case OSC_LEVEL:
    if (first != OSC_LEVEL) /*if not sweep variable*/
      *val=HP_getpar("OSC?");
    else
      *val=HP_read(2);
     break;
case OSC_FREQUENCY:
   if (first != OSC_FREQUENCY)
     *val = HP_getpar("FREQ?");
   else
     *val = HP_read(2);
   break;
/*************************************************************/
/****  Get cable switch info                          ****/
/*************************************************************/
case HP_SWITCH:
   *val = SWITCH_get();
    error = ((*val < -.9) ? -1 : 0);
    break;
case HP_REAL:
   *val = HP_read(0);
   if (*val == ESCAPE)
     error = ESCAPE;
   break;
case HP_IMAGINARY:
   *val=HP_read(1);
   if (*val == ESCAPE)
     error = ESCAPE;
   break;
case TOTALPAR-1:
    error = 0;
    *val = s_time();
    break;
/*************************************************************/
/***   Get DC Voltage/Resistance                 ********/
/*************************************************************/
case TOTALPAR:
case TOTALPAR+1:
  error = 0;
  if (channel == 0)  {
    *val = 0;
    wait_plus(s_time()+1000*(long)(hold+.999),0);
    *val = NVMETER_volts (&a);
    if (l == TOTALPAR+1)
      *val/= current[0];
```

```
  }
  else{
    CURRENT_standby();
    SCNNER_reset();
    SCNNER_channel(SCNNER_CLOSE,channel);
    SCNNER_channel(SCNNER_CLOSE,channel+CURRENT_CHANNEL_OFFSET);
    SCNNER_channel(SCNNER_DISPLAY,channel);
  }
  a = 0;
  tvolt[0] = tvolt[1] = 0.0;
  for  (j = 0; j <= 1 && a > -1; j++)  {
    CURRENT_operate(3*channel+1+j);
    if (options[NV_FILTER] >= 0)  {
      wait_plus(s_time()+(long)1000*options[NV_FILTER],0);
      NVMETER_set_par (-1, -1, -1, -1, 2);
    }
    if (options[NV_DAMPING] >= 0)  {
      wait_plus(s_time()+(long)1000*options[NV_DAMPING],0);
      NVMETER_set_par (-1, -1, 1, -1, -1);
    }
    m = wait_plus(s_time() + (long)1000 * hold, (hold < 1) ? -100 : 1);
    if (m == 0x11B)  {
      CURRENT_standby();
      SCNNER_reset();
      SCNNER_channel(SCNNER_CLOSE,SCNNER_SHORT);
      return(m);
    }
    if (options[NV_STABLIZE] > 0 && options[NV_WAIT] > 0
       && options[NV_ERROR] >= 0) {
     time0 = s_time();
     while ((um = tvolt[j] = NVMETER_stable (time0,options[NV_WAIT],
           options[NV_ERROR] * 1e-9,options[NV_STABLIZE])) > 0 && um < 0xFFFF) {
       if (um == 0x11B)  {
         CURRENT_standby();
         SCNNER_reset();
         SCNNER_channel(SCNNER_CLOSE,SCNNER_SHORT);
         return um;
       }
       else if (um == 0x1C0D)
         break;
       else
         push(um);
     }
    }
    else
      tvolt[j]=NVMETER_volts(&a);
    CURRENT_standby();
  }
  if (a > -1) {
    SCNNER_reset();
    SCNNER_channel(SCNNER_CLOSE,SCNNER_SHORT);
    *val = (tvolt[0] - tvolt[1]) / (2.0);
    if (l == TOTALPAR + 1)  {
     if (current[channel] != 0)
```

```
                *val/= current[channel];
              else
                *val=0;
            }
          }
          else  {
            error = -1;
            printf("subroutine done nv_stat=%i\n",a);
          }
          break;
        default:
          break;
        }
        if ((m = _bios_keybrd(_KEYBRD_READY)) != 0)  {
          if (m == 0x11B || m == 0x1C0D)
            return _bios_keybrd(_KEYBRD_READ);
          else
            push(_bios_keybrd(_KEYBRD_READ));
        }
        return error;
}

/**** set up keyboard buffer as a "stack"                ****/
/**** actually, it is a first on - first off buffer      ****/
/**** The first values entered will be overwritten if    ****/
/**** you go beyond the stack size.                      ****/

#define STACK_SIZE 15
static unsigned stack[STACK_SIZE];
static int stack_loc;
static first_loc;

int reset_stack()
{
        stack_loc = 0;
        first_loc = 0;
}

int push (unsigned val)
{
        stack[stack_loc] = val;
        stack_loc = (stack_loc + 1) % STACK_SIZE;
        if (stack_loc == first_loc)
          first_loc = (first_loc + 1) % STACK_SIZE;
        return 0;
}

unsigned pull()
{
        unsigned temp;

        if (first_loc == stack_loc)
```

```
        return 0;
      temp = stack[first_loc];
      first_loc = (first_loc + 1) % STACK_SIZE;
      return temp;
}

/************************************/
/* branch(): converts angle to a specific range.
/* This range is centered at center and size.
/* Parameters:
/*     value - angle to be converted
/*     center - center of branch range (e.g. might be 0 to give range -180 to +180 degrees)
/*     size - size of branch range (e.g. would be 360 for degrees)
/* Return: new angle within specified range
/************************************/
double branch(value, center, size)
double value, center, size;
{
      center = (center - value) / size - .5;
      if (center <= value || (int)center == center)
        value += size* (int)center;
      else
        value += size * (int)(center + 1);
      return value;
}

static long start_time;
/*********************/
/*  init_time(): resets the system time to 0 (program's counter, not the computer clock)
/*  this also resets the keyboard stack (just because everyone would forget to call it)
/***************/
void init_time()
{
      struct timeb timebuffer;

      ftime(&timebuffer);
      start_time = timebuffer.time;
}

/************/
/*s_time(): returns a number corresponding to the system time
/*     This system time is used by ALL routines which need to know a time.
/*     It is the real time in msec since the Init_time() routine was called
/*********************/
long s_time()
{
      struct timeb timebuffer;
      long ttime;

      ftime(&timebuffer);
      ttime = (timebuffer.time - start_time) * 1000 + timebuffer.millitm;
      return ttime;
```

```
}

long f_time()
{
        struct timeb timebuffer;
        long ttime;

        ftime(&timebuffer);
        return timebuffer.time;
}

/*************************************************************************/
/*************************************************************************/
/*liquid nitrigen filling control
/*************************************************************************/

static long last_fill;
static int meter_addr;
int LN2_reset(c)
int c;
{
        last_fill = f_time();
        /* decide what meter is reading the fill */
        if (NVMETER_read2(6) > 1)  {
          meter_addr = 6;
          printf("LN2 is read from 181 at address 6");
        }
        else if (NVMETER_read2(22) > 1)  {
          meter_addr = 22;
          printf("LN2 is read from 199 at address 22");
        }
        else {
          printf("ERROR---no voltmeter avalilble\n assume address 22");
          meter_addr = 22;
        }
        LN2_off();
        printf("initialized LN2 fill\n");
        if (c == 'N'|| c == 'n')
         last_fill -= FILL_INTERVAL;    /* force fill now */
}

int LN2_fill()
{
        int n,m;
        double below,level;
        int abort;

        abort = !TRUE;
        if (f_time() > last_fill + FILL_INTERVAL )  {   /* if time to fill again */
         below = ((FULL_SIGNAL > 0) ? FULL_SIGNAL - FULL_STEP :
                        NVMETER_read2(meter_addr));
         printf("FILLING LN2");
```

```
    LN2_on();
    for (n = 0; n < 120 && abort  != TRUE; n++)  {  /* wait for initial splashing to calm down */
       if (FULL_SIGNAL <= 0)  {
         if (fabs((level = NVMETER_read2(meter_addr)) - below) < FULL_STEP / 5)
           below=level;
       }
       if (wait_plus(s_time() + 1000,2) == ESCAPE)
         abort = TRUE;
    }
    n = 0;
    m = 0;
    while ( !abort && n < 10 && m < 900)  {    /* never fill more than 15 min */
       printf("-");
    if (wait_plus(s_time() + 1000, 2) == ESCAPE)
     abort = TRUE;
    while ((level = NVMETER_read2(meter_addr)) > below+FULL_STEP && n < 10 )  {
       if (wait_plus(s_time() + 2000, 2) == ESCAPE)
         abort = TRUE;
       n++;
     }
     if (FULL_SIGNAL < 0)  {
      if (n == 0 && fabs(level-below) < FULL_STEP / 10)
        below = level;
     }
     if (n < 10)
      n = 0;
      m++;   /* keep track of total fill time */
  }
  LN2_off();
  last_fill = f_time();
  printf("\n Fill done- wait 3 min");
  if (!abort)
    wait_plus(s_time() + (long)120 * 1000, 2);
}
```

A.3 Device Routines

This section contains the third sublevel of routines described in Chapter 4.8.2 that control the communication to specific devices. Included here are the routines used to control a few of the devices used including the HP4194A impedance analyzer and the LakeShore 93C temperature controller.

A.3.1 HP4194A Control Routines

Listing of File Mhp.c

```
#include      "C:\lw\include\lwsystem.h"
#include      "C:\lw\include\gpib.h"
#include      "C:\lw\include\formatio.h"
#include      "C:\lw\include\optionh.h"
#include      "C:\lw\include\temp_def.h"
#include <bios.h>
#include <string.h>
#include  <stdio.h>
#include <sys\timeb.h>
#include <time.h>
#include <graph.h>
#include <color.h>
#include <math.h>

#define MAX_FILE 40
#define TO_RAD 1.745329252e-2

/*** LOCAL VARIABLES **********/
extern int options[MAX_OPTIONS];
static int hp_addr = -1;   /* stores IEEE address for HP */
static int sweeping;    /*used to tell if scan is in progress*/
static union hp_dat  { /*used to convert HP float format to C format*/
      double flt;
      char chr[8];
} hp_data[3][402];
/*****************************************************/
/* init () - initializes HP4194A and sends initialization string */
/*return: -1 on error, address if successful*/
int HP_init (int addr)
{
      char cmd[30];
      sweeping = 0;
      hp_addr = addr;
      if (GEN_init(addr) < 0)
        return -1;
      return addr;
}
```

```
/*****************************************************************/
/*This instrument removes the instrument from the GPIB device table and */
/*sets nvmeter_addr to zero.                                      */
/*return: -1 on error, 0 if successful                            */
/*****************************************************************/
int HP_close()
{
    if (GEN_loc(hp_addr) < 0)
      return -1;
    hp_addr = -1;
    return 0;
}
/*****************************************************************/
/*HP_ready: wait for end of HP sweep
/*Parameters:
/*        time0: time wait was started- reference time for timeout limit
/*        timeout: time in seconds to wait for HP before giving up
/*Return: 0 if scan is done, -1 on timeout limit, or key code if any key is pressed
/*NOTE: HP_ready will only work after a call to HP_start_sweep. If HP_start_sweep has
/*        not been called, HP_ready will immediately return 0.          */
/*NOTE: During the wait, this routine will record the temperatur fluctuations*/
int HP_ready(long time0, int timeout)
{
    extern double temp_min[3];
    extern double temp_max[3];
    extern long temp_num[3];
    extern double temp_sum[3];
    extern double temp_sqr[3];
    long timeout2;
    int n, m, t;
    int first;
    double ttemp;

    first = 0;
    if (sweeping == 0)  /*If HP_start_sweep has not been called, return 0*/
      return 0;
    GEN_write(hp_addr,"RQS2",4);
    timeout2 = (long)timeout * 1000;
    if (((m = GEN_rsp(hp_addr)) & 66) == 66) /*Check SRQ to see if HP is active*/
      n = 0;
    else
      n = -1;
    while (n == -1 && s_time() - time0 < timeout2)  {
       if ((keypress = wait_plus(s_time() + 1000, 1)) != 0)/*return on key press*/
         return keypress;
       for (t = 0; t <= 2; t++)  {
          if ( t< 2)
            ttemp = TEMP_get_temp(t,&first);/*keep track of temperature readings*/
          else
            ttemp = TEMP_power();/*keep minimum and maximum values*/
          if (ttemp > 0)  {
            temp_sum[t] += ttemp;
            temp_sqr[t] += ttemp * ttemp;
```

```
            if (temp_max[t] < ttemp)
              temp_max[t] = ttemp;
            if (temp_min[t] > ttemp)
              temp_min[t] = ttemp;
            temp_num[t]++;
          }
        }
        if (((m = GEN_rsp(hp_addr)) & 66) == 66)
          sweeping = n = 0;
        else
      }
      if (n == -1)
        beep();
      return n;
} */
/*************************************************************************/
/*HP_sweep: set up sweep on the HP. This only sets the sweep parameters. It
/*        does not start the sweep (see HP_start_sweep())
/*Parameters:
/*        type: string corresponding to type of sweep (sweep frequency, osc. level, etc.)
/*                (normally sweep type is set from HP front panel, so this parameter is
/*                not needed).
/*        start: start value for sweep
/*        end: end value for sweep
/*        nop: number of points in sweep
/*        mult: 0 for log sweep, !=0 for linear sweep
/*        delay: delay(seconds) between each reading in the sweep
/*Return: always 0
/*************************************************************************/
int HP_sweep(char *type, float start, float end, int nop, int mult, double
delay)
{
    /*printf("setting up sweep %s from %f to %f nop=%i, mult=%i\n",type,start,end,nop,mult);*/

    GEN_write(hp_addr,type,strlen(type));
    HP_set("START=", (start < end) ? start: end);   /* sweep up:down*/
    HP_set("STOP=", (start < end) ? end : start);    /* sweep up:down*/
    if (nop < 1)
        nop = 1;
    HP_set("NOP=", (float)nop + 1);                 /* number of points*/
    GEN_write(hp_addr, (mult == 0) ? "SWT2" : "SWT1", 4);   /* log:linear */
    GEN_write(hp_addr, (start > end) ? "SWD2" : "SWD1", 4); /* sweep down :up*/
    HP_set("DTIME=", (float)delay * 1000);
    return 0;
}
/*************************************************************/
/*HP_set: used to send arbitray command to the HP
/*        cmd: HP variable to set (see reference manual)
/*        val: Value to set variable to
/*Return: none*/
int HP_set(char *cmd, float val)
{
```

```
    char temp[30];
    sprintf(temp,"%s%20.10E",cmd,val);
    return GEN_write(hp_addr,temp,strlen(temp));
}
/*****************************************************************/
/*HP_set2: used to send arbitray command to the HP with units
/*       cmd: HP variable to set (see reference manual)
/*       val: Value to set variable to
/*       unit: Unit to use for value (e.g. Hz, MHz)
/*Return: none*/

int HP_set2(char *cmd, float val, char *unit)
{
    char temp[30];
    sprintf(temp,"%s%.10E%s",cmd,val,unit);
    return GEN_write(hp_addr,temp,strlen(temp));
}
/*****************************************************************/
/*HP_read: read sweep value from the HP. This returns a single value from the
/*       array read during the sweep.
/*Parameters::
/*       col: column to read (which variable)
/*       row: row to read (which step in the sweeep)
/*Return: value at the corresponding column,row
/*NOTE: a call with col=row=0 causes all values to be transferred from the HP
/*       to the computer. Only after this is done will valid numbers be read for
/*       all array elements. Successive calls to HP_read can be used to get all
/*       measured values.*/
double HP_read(int col, int row)
{
    unsigned char cmd[30];
    char *names;
    int n, m, num;

    if (row > 400)
      row = 400;
    if (col >= 3 || col < 0)
      col = 2;
    if (row <= 0) {
      wait_plus(1000 + s_time(), 0);
      row = 0;
      names = "ABX";
      GEN_write(hp_addr,"FMT2",4);
      *cmd = names[col];
       cmd[1] = '?';
       cmd[2] = 0;
       GEN_write (hp_addr,cmd,2);
       GEN_read(hp_addr,cmd,4);
       num = (cmd[2] * 256 + cmd[3]) / 8;
       for (n = 0; n < num; n++) {
          GEN_read(hp_addr,cmd,8);
          for (m = 0; m < 8; m++)
             hp_data[col][n].chr[m] = cmd[7-m];
```

```
        }
      }
      return hp_data[col][row].flt;
}
/*********************************************************/
/*HP_start_sweep: Tells HP to start the sweep that has been set up. See also
/*        HP_sweep() and HP_ready().
/*Parameters: none
/* Return: none
int HP_start_sweep()
{
      sweeping = 1;
      GEN_write(hp_addr,"AUTOA",5);
      GEN_write(hp_addr,"AUTOB",5);
      return  GEN_write(hp_addr,"SWTRG",5);
}
/*********************************************************/
/*HP_scale: set scale on HP screen
/*Parameters:
/*        channel: which channel to rescale channel=6 is 'A', any other
/*                 channel is 'B'
/*        scale: I don't know. I just see it rescale the screen
/*RETURN: none
int HP_scale(int channel, int scale)
{
      char cmd[30];
      char chan;
      int n;

      n = (scale == 1) ? 1 : 2;
      chan = (channel == 6) ? 'A' : 'B';
      GEN_write(hp_addr,"DSP1",4);
      sprintf(cmd,"%cSC%i",chan,n);
      return GEN_write(hp_addr,cmd,4);
}
/*****************************************************************/
/*HP_getpar: Reads the value of a specific HP variable. See also HP_set().
/*Parameters
/*        cmd: Name of HP variable to read
/*Return: value contained in HP variable.*/
float HP_getpar(char *cmd)
{
      char cmd2[30];
      int m;
      union hp_dat{
            double flt;
            char chr[8];
      } data;

      GEN_write(hp_addr,"FMT2",4);
      GEN_write (hp_addr,cmd,strlen(cmd));
      GEN_read(hp_addr,cmd2,4);
```

```
    GEN_read(hp_addr,cmd2,8);
    for (m = 0; m < 8; m++)        /*Convert form HP float format to C format*/
       data.chr[m] = cmd2[7-m];
    return data.flt;
}
```

A.3.2 Temperature Controller Routines

```
/* A series of Subroutines used for controlling the Lakeshore 91C and 93C Temperature
/* controllers. These subroutines are used to set the temperature, read the temperature
/* and wait for the temperature to stabilize.*/

/* Note that references to high_res refer to an option to allow the computer to control the
/* Sample temperature to ±0.001K */

#include      "C:\lw\include\lwsystem.h"
#include      "C:\lw\include\gpib.h"
#include      "C:\lw\include\formatio.h"

#include      <stdio.h>
#include      <bios.h>
#include      <math.h>
#include "nrutil.h"
#include      "scan_def.h"

/*addresses used for high res temperature reading*/
#define NV_ADDRESS 6       /*IEEE Address of voltmeter used for high_res sensor reading*/
#define CUR_ADDRESS 12     /*IEEE Address of current source used for high_res sensor*/
#define CURRENT 50e-6      /*current(A) applied to read high res sensor resistance*/
#define INTER_RES 3        /*number of points for interpolation */


/* global variable definitons */
extern float tc_time,tc_gain,tc_reset,tc_rate,tc_max_power;  /*Parameters for high_res control*/
extern double tc_error, tc_max_wait,tc_stable;  /* Allowed error for high_res control*/
extern int options[MAX_OPTIONS];   /* Contains settings for temperature control parameters*/

static int temp_addr = -1;       /* temp_addr contains the descriptor returned by OpenDev */
static int high_res = FALSE;  /*Determines whether or not to use high_res control*/

/*****************************************************************/
/* init () - initializes temperature controller and sends initialization string */
/*         if global variable int HIGH_RESOLUTION is TRUE, initialize and use
/*         high resolution sensor control. See gscan.c for global variable list*/
/* Parameters:
/*     addr: IEEE address of temperature controller
/*     high_res: TRUE-use high resoluction sensor
/*               FALSE - do not use high resolution sensor*/
/* return: -1 on error, address if successful*/
/*****************************************************************/
int TEMP_init (int addr, int par_high_res)
{
     high_res = par_high_res;
```

```
    if (GEN_init(addr) < 0)  {
       printf ("ERROR: TEMP_init(%i)\n",addr);
       return -1;
    }
    if (addr > 0)
     if (GEN_write (addr,"M1T3Z0\r\n", 8) <= 0)  {
       printf ("error: TEMP_init(%i)=%x\n",addr,iberr);
       return -1;
     }
    if (high_res && TEMP_high_init() < 0)   /*will not do high_int if high_res==FALSE */
      printf("error: TEMP_init(%i) high resolution calibration not loaded\n",addr);
    temp_addr = addr;
    return temp_addr;
}
/*********************************************************************/
/*This instrument removes the instrument from the GPIB device table and*/
/*sets temp_addr to zero.*/
/*return: -1 on error, 0 if successful*/
/*********************************************************************/
int TEMP_close ()
{
    int n;
    n = 0;
    /*  Check for device closed */
    if (temp_addr <= 0)
      return -1;
    /*  Close the device    */
    GEN_write(temp_addr,"M0",2);
    if (GEN_loc(temp_addr) < 0)
      n = -1;
    temp_addr = 0;
    return n;
}

/*********************************************************/
/* TEMP_clear_sqr() -clears service request buffer.  This removes
/*     any old srq messages so only new ones will be seen.
/*     This routine should be called before waiting for a new message.
/* Return: -1 on error, 0 if succesful
/* Note: If there is a communication problem with the device
/*     this routine could hang up
/*********************************************************************/
int TEMP_clear_srq ()
{
    int srq;
    if (temp_addr <= 0)    /* if not initialized, return error*/
      return -1;
    srq = 1;
    while (srq != 0)
        srq = GEN_rsp (temp_addr);
    return 0;
}
```

```
/*********************************************************************/
/*TEMP_setpoint : sets setpoint on temperature controller
/*Parameter:
/*      setpoint = value to use for new setpoint
/*Return: <0 on error
/*********************************************************************/
int TEMP_setpoint (float setpoint)
{
      char temp[30];
      int m, n;
      m = -1;
      n = 0;

      if (temp_addr <= 0)   /* if not initialized, return error*/
        return -1;
      sprintf (temp, "S%f", setpoint);  /*create command string to send to controller */
      return GEN_write(temp_addr,temp,strlen(temp));
}

/*********************************************************************/
/*TEMP_get_temp - reads temperature from the controller
/* Parameter: channel
/*      channel<0 : get setpoint temperature
/*      channel=0 : get control sensor
/*      channel>0 : read sample sensor.  If scanner is used, set scanner
/*            channel to (channel-1).
/*Return value: Temperature value
/*********************************************************************/
double TEMP_get_temp (int channel)
{
      double temp;
      int n;
      char temp2[80];

      if (temp_addr <= 0)  /* if not initialized, return error*/
        return -1;

      *temp2 = 'W';         /*Build command string to request the value from controller*/
      if (channel < 0)
        temp2[1] = 'P';
     else if (channel == 0)
       temp2[1] = 'C';
    else {
      TSENSE_channel (channel-1);
      TSENSE_command ("N");
      wait_plus(s_time()+(long)1000,-200);
      temp2[1] = 'S';
    }
    if (GEN_write(temp_addr,temp2, 2) <= 0)   /*send command to controller */
      return -1.0;
    if (GEN_read(temp_addr,temp2, 10) <= 0)   /*Read value from controller */
      return -1.0;
    if (sscanf (temp2, "%lf", &temp) != 1)      /*Convert value from ASCII to double*/
```

```
    return -1.0;
  return temp;
}

/*************************************************************************/
/*TEMP_stable : wait for temperature to stabilize
/*        This routine calls TEMP_at_setpoint or TEMP_high_control
/*        wait will be aborted with either enter or escape
/*Parameters :
/*    external variable options[] contains all stabilization values
/*    high_res determines whether this will be stabilization on
/*    the Lakeshore controller or high-res controller
/*RETURN:
/*    0xFFFF on timeout (if it did not stabilize after maximum wait time)
/*    0 if temperature is stable
/*    key code for ESCAPE or ENTER if either key is pressed
/*NOTE: if any other key is pressed, it is pushed onto a stack - see push() function
/*    in file mscnsub.c
/*************************************************************************/
unsigned TEMP_stable()
{
    unsigned long time1;
    double p_error, n_error;
    unsigned um;

    if (!high_res) {          /*if using Lakeshore controller*/
      if (options[TC_TRAIL_ERROR] < 0){  /*Calculate allowed errors*/
        p_error = 1e6;
        n_error = -1e6;
      }
      else {
        p_error =options[TC_LEAD_ERROR] + options[TC_TRAIL_ERROR]) * .001;
        n_error = (options[TC_LEAD_ERROR] - options[TC_TRAIL_ERROR]) * .001;
      }
      time1 = s_time();
        /*Wait until temperature is within allowed errors*/
      um = TEMP_at_setpoint(time1,options[TC_WAIT], -500.0,
            n_error,p_error,options[TC_STABLIZE]);
    }
    else {                  /* if using high res temperature control */
      if (tc_error < 0) {
        p_error=1e6;
        n_error=-1e6;
      }
      else {
        p_error = options[TC_LEAD_ERROR] * .001 + tc_error;
        n_error = options[TC_LEAD_ERROR] * .001 - tc_error;
      }
      time1 = s_time();
        /*Wait until temperature is within allowed errors*/
      um = TEMP_high_control(time1,tc_max_wait,-500.0,n_error,perror,tc_stable);
    }
    return um;
```

```
}

/*****************************************************************************/
/*TEMP_at_setpoint : wait for temperature to stabilize (also return if key is pressed)
/*Parameters:
/*     time0:  start time (usually initialized to s_time() just before calling this routine)
/*     timeout: maximum time in seconds to wait after time0 before returning.
/*          At s_time()=time0+1000*timeout, return from routine even if temperature is not stable
/*     sp : setpoint to stabilize at. if sp<-490, read setpoint from controller
/*     n_error: must have (temperature>sp+n_error) to be considered stable
/*     p_error: must have (temperature<sp+p_error) to be considered stable
/*     stable: time to wait with (sp+n_error<temperatur<sp+p_error) before temperature
/*          is really considered stable and return from routine
/*Return:
/*     -1 on error
/*     0 if temperature is stable
/*     key code if enter or escape was pressed
/*****************************************************************************/
unsigned TEMP_at_setpoint (unsigned long time0,int timeout,double
sp,double n_error,double p_error,int stable)
{
     unsigned keypress;
     long timeout2;
     int n;
     long stable2;
     long ctime;
     long time1;
     double setpt;
     double contrl;
     int zero;
     zero = 0;
     stable2  = (long)stable * 1000;
     timeout2 = (long)timeout * 1000;
     setpt = (sp < -490) ? TEMP_get_temp (-1) : sp;
     contrl = TEMP_get_temp (0);
     n = -1;
     ctime = s_time();
     while (n == -1 && ctime - time0 < timeout2)  {
        if (contrl <= setpt + p_error && contrl >= setpt + n_error)  {
          time1 = ctime;
          n = 0;
          while ( n == 0  && ctime - time1 < stable2)  {
             /* temperature is within error wait to make sure it stays there*/
            if ((keypress = wait_plus(s_time() + 1000, 1)) != 0)
              return keypress;
            ctime = s_time();
            contrl = TEMP_get_temp (0);
            zero = 0;
            setpt = (sp < -490) ? TEMP_get_temp (-1) : sp;
            zero = 0;
            if (contrl > setpt + p_error || contrl < setpt + n_error)
              n = -1;
          }
```

```
        }
        if (n < 0)  {
          if ((keypress = wait_plus(s_time() + 1000, 1)) != 0)
            return keypress;
          ctime = s_time();
          contrl = TEMP_get_temp (0);
          zero = 0;
          setpt = (sp < -490) ? TEMP_get_temp (-1) : sp;
          zero = 0;
        }
      }
      if (n < 0) {
        beep();
        return 0xFFFF;
       }
      return  n;
}

/*************************************************************************/
/*************************************************************************/
/* The following routines are for the high-resolution temperature control
/*************************************************************************/
/*************************************************************************/
double setpt = -1;      /* set as invalid until temp_high_control is called */
double old_error;
double out_current;
double contrl;          /*actually a 'return' value from wait_plus() */
double contrl_avg;
int contrl_num;

/*************************************************************************/
/*TEMP_pow_init -reads present output current and starts from there for temperature control*/
/* returns 0
/*************************************************************************/
int TEMP_pow_init()
{
      out_current = pow(MAGNET_read_current(),2);
      return 0;
}
/*************************************************************************/
/* TEMP_high_setpoint : sets setpoint value for high res temperature control
/*      It also initializes output current to the present value
/*Parameter: sp
/*      sp=new setpoint (K)
/* returns 0
/*************************************************************************/
int TEMP_high_setpoint(double sp)
{
      setpt = sp;
      old_error = 0.0;
      out_current = pow(MAGNET_read_current(),2);
      return 0;
}
```

```
/*********************************************************************/
/*TEMP_high_control : wait for temperature to stabilize (also return if key is pressed)
/*    NOTE: all connection with LIA has been removed--see LIA_auto_wait().
/*Parameters:
/*    time0:  start time (usually initialized to s_time() just before calling this routine)
/*    timeout: maximum time in seconds to wait after time0 before returning.
/*          At s_time()=time0+1000*timeout, return from routine even if temperature is not stable
/*    sp : setpoint to stabilize at  (set setpoint to this value)
/*    n_error: must have (temperature>sp+n_error) to be considered stable
/*    p_error: must have (temperature<sp+p_error) to be considered stable
/*    NOTE: this routine really just waits until temperature fluctuation is
/*          <|p_error-n_error| regardless of where the temperature is really stabilizing
/*    stable: time to wait with (sp+n_error<temperatur<sp+p_error) before temperature
/*          is really considered stable and return from routine
/*Return:
/*    -1 on error
/*    0 if temperature is stable
/*    key code if key was pressed
/*********************************************************************/
unsigned TEMP_high_control (unsigned long time0,int timeout,double
sp,double n_error,double p_error,float stable)
{
    long timeout2;
    int n;
    long stable2;
    double t_width;
    FILE *output;
    unsigned keypress;     /* read keys that might be pressed */
    long end_stable;
    double contrl_min, contrl_max;
    double max_err;
    long ctime;

    TEMP_get_high_init();   /* reset temperature averaging */
    output = fopen("t_p_time.dat","w");
    out_current = pow(MAGNET_read_current(),2);
    printf("\nInitial out_current=%f\n",sqrt(out_current));
    stable2 = (long)stable*1000;
    timeout2 = (long)timeout*1000;
    if (sp > 0)
      setpt = sp;
    if (setpt < 10)  {
      printf("ERROR!!!! setpoint not set\n");
      beep();
    }
    GEN_timeout(13);
    contrl = TEMP_get_high();
    /*Use plot of temperature vs. time to optimize control*/
    cls ();
    DeletePort(0);
    SetTitle ("Waiting for Stabilization");
    SetFrmColor (2);
```

```
SetTtlColor (15);
SetGrdColor (2);
SetLblColor (2);
SetBckColor (0);
SetAxName (0, "Time");
SetAxScale (0, 0);
SetAxAuto (0, 0);
SetAxRange (0, (double)time0, (double)(time0+timeout2), 10);
SetAxName (1, "Temperature");
SetAxScale (1, 0);
SetAxAuto (1, 0);
t_width = 1.5 * fabs(contrl - setpt) + 2 * fabs(p_error - n_error);
SetAxRange (1,setpt-t_width, setpt+t_width, 10);
max_err = fabs(p_error-n_error);          /*maximum allowed error*/
contrl_num = 0;
contrl_avg = 0;
contrl_min = contrl_max = contrl;
end_stable = s_time() + stable2;
n = 0;
while (s_time() < end_stable)  {
    /* Read temperatur, control, and wait a maximum of .5 sec */
    if ((keypress = wait_plus(s_time()+500,1)) != 0)
      return keypress;
    if (contrl > contrl_max)    /*keep track of maximum and minimum temperatures*/
      contrl_max = contrl;
    if (contrl < contrl_min)
      contrl_min = contrl;
    if (contrl_max-contrl_min > max_err){
      contrl_max = contrl_min = contrl;    /*reset stabilizing conditions*/
      contrl_num = contrl_avg = 0.0;
      SetCrvColor(10);
      end_stable = s_time() + stable2;
      if (end_stable > time0 + timeout2)  {   /* if at end of timeout*/
        end_stable = time0 + timeout2;
        n = -1;
      }
      else
        n = 0;
    }
    else {
      contrl_avg += contrl;
      contrl_num++;
      SetCrvColor(12);
    }
    ctime = s_time();
    SetPlotMode (0);
    SetPointStyle (5);
    GrfPoint2D ((double)ctime,(double)contrl);
    fprintf(output,"%f %f %.8g\n",(float)ctime,(float)contrl,out_current);
}
fclose(output);
if (n < 0) {
  beep();
```

```
        return 0xFFFF;
      }
      return  n;
}

/*************************************************************************/
/*TEMP_high_avg : returns an average temperature.  If temperature was controlling, this
/*     is the average temperature while temperature was stable.
/*     If there was no stabilization time, it reads the present temperature
/*     and returns
/*Return value : temperature (K)
/*************************************************************************/
double TEMP_high_avg()
{
        if (contrl_num > 0)
          return contrl_avg / contrl_num;
        else
          return TEMP_get_high();
}

/*************************************************************************/
/*waid_plus() : subroutine can be called anytime from anywhere and must be able to */
/*control the high resolution temperature*/
/*     NOTE: all keys pressed will be pushed onto the stack using push()
/*          EXCEPT if key=-1.  For key=-1, all values are just returned.
/* ret_time=time to return (using s_time())*/
/* key=0 if ignore keypress*/
/*   =-1 if return on any key */
/*   =1 if return on ESCAPE and ENTER */
/*   =2 if return only on ESCAPE     */
/*   =-100  Time is critical -- do nothing but return on time */
/* GLOBAL VARIABLES: (from TEMP_high_control) */
/* double out_current = present output power to heater */
/* double setpt = present temperature set point */
/* double old_error = previous setpt-control */
/* double contrl = must return TEMP_get_high to control program */
/* Note : don't try to control temperature before setpt is properly defined */
/*************************************************************************/
long next_read;          /* keeps track of when to update heater next */
int wait_init()
{       /* initialize variables (called from init_time()) */
     next_read  = 0;
}

#define CIM_K 0.01065
unsigned  wait_plus(long ret_time, int key)
{
     unsigned long time1;
     unsigned keypress;

     time1 = s_time();
     GEN_timeout(13);     /* set timeout=10 sec */
```

```
    /*in case wait_init was never called and there is an error in next_read*/
    if (next_read > time1 + (long)tc_time)
      next_read = 0;

    do {
       /*Check for keys that are pressed */
       if (key > -100 && key != 0 && _bios_keybrd(_KEYBRD_READY) != 0)  {
          keypress =  _bios_keybrd(_KEYBRD_READ);
          if (keypres == ESCAPE || key != -1)
            push(keypress);       /*save key for future use*/
          if (key == -1 || keypress == ESCAPE || (key == 1 && keypress == ENTER))
            return keypress;
       }
       /* if using high res temperature control */
       if (key  > -100 && setpt > 0 && time1 > next_read && high_res)  {
         /*Read temperature and adjust heater output accordingly */
         contrl = TEMP_get_high();
         out_current += tc_time * (tc_gain * (setpt-contrl-old_error) + tc_reset*(setpt-contrl));
         old_error = setpt-contrl;
         if (out_current > tc_max_power * tc_max_power * (CIM_K * CIM_K))
           out_current = tc_max_power * tc_max_power * (CIM_K * CIM_K);
         else if (out_current < 0)
           out_current = 0;
         MAGNET_set_current(sqrt(out_current));
         next_read = time1+ (long)tc_time;
       }
    } while (ret_time > (time1= s_time()));
    return 0;
}

/*****************************************************************/
/*****************************************************************/
/*The following routines are used to read high resolution sensor and
/*use a calibration table to convert between resistance and temperature
/*****************************************************************/
static float res_cal[2][251];      /* data for calibration of CGR sensor */
static unsigned long num_cal = 0;
static unsigned long point = 1;

/*********************************************************************/
/*TEMP_high_init - read calibration table and initialize variables for
/*      reading high resolution sensor
/*RETURN: number of points read in the calibration table
/*********************************************************************/
int TEMP_high_init()
{
    FILE *cal_dat;
    num_cal = 0;
    init_time();              /* must initialize timer for wait */
    wait_init();

    TEMP_pow_init();
    if ((cal_dat = fopen("C:\\lw\\programs\\fscan2\\cgr__cal.dat","r")) == NULL)
```

```
        return -1;
      do {
         if (fscanf(cal_dat,"%f%f",&(res_cal[1][num_cal]),&(res_cal[0][num_cal])) > 1) {
           printf("%f %f\n",res_cal[0][num_cal],res_cal[1][num_cal]);
           num_cal++;
         }
      } while(!feof(cal_dat));
      fclose(cal_dat);
      printf("read %i points\n",num_cal);
      return (int)num_cal;
}

#define AVERAGE 15          /*number of temperature points to average over */
float old_temp[AVERAGE];    /* buffer to hold previous temperature readings*/
int cur_temp = -1;

int TEMP_get_high_init()
{
      cur_temp = -1;
}

int TEMP_high_reset()
{
      double cur;
      wait_init();
      NVMETER_init(NV_ADDRESS);
      GEN_clr(CUR_ADDRESS);
      cur = CURRENT_read2(13);
      GEN_clr(13);
      CURRENT_set2(13,cur);
}

/*************************************************************************/
/*TEMP_get_high : read present temperature and convert it from resistance to K
/*Return: -1 on error
/*      temperature (K) measured by high resolution sensor
/*************************************************************************/
double TEMP_get_high()
{
      float volts, temp, error;
      int k, count;
      float offset;

      if (cur_temp < 0 || cur_temp >= AVERAGE)  {
        for (k = 0; k < AVERAGE; k++)
           old_temp[k] = -1;
        cur_temp = 0;
      }
      offset = 0;
      old_temp[cur_temp] = NVMETER_read_volt(NV_ADDRESS,CUR_ADDRESS,0,
                           (double)CURRENT)/CURRENT - offset;    /* read voltage */
      cur_temp = (cur_temp + 1) % AVERAGE;
```

```
    volts = count = 0;
    for (k = 0; k < AVERAGE; k++)  {
       if (old_temp[k] > 0)  {
         volts += old_temp[k];
         count++;
       }
    }
    if (count > 0)
      volts /= count;
    else
      return 0.0;
    hunt(res_cal[0]-1,num_cal,volts,&point);   /*Get number from an interpolation table*/
    k = IMIN(IMAX(point - (INTER_RES-1) / 2, 1),num_cal + 1 -INTER_RES);
    polint(&res_cal[0][k-1]-1,&res_cal[1][k-1]-1, INTER_RES,volts,&temp,&error);
    return (double)temp;
}
```

A.4 General Interface Routines

The following set of routines is the low level device interface routines as described in Chapter 4.8.2. These routines are responsible for transferring ASCII command strings to the electronic instruments and receiving the ASCII data strings from these instruments. The actual data transfer occurs in the routines ibwrt() (interface board write) and ibrd() (interface board read), but an equally important routine is ibcmd() (interface board command) which sends commands to the GPIB interface board installed in the computer. The various calls to ibcmd() instruct the GPIB board how to control the flow of data between the computer and the remote devices by setting the address of the device which is to send or receive the ASCII string. The most commonly used of these "GEN" routines are GEN_write() and GEN_read() which are specifically responsible for sending (writing) and receiving(reading) the ASCII strings. Another important routine is the device initialization routine GEN_init() which helps to transfer devices from the local mode where the device is controlled by the front panel, to the remote mode in which the device is controlled by the computer. Although most devices adjust to this transfer automatically and do not require this initialization routine, some devices (most often the EG&G lock-in amplifiers) require explicit instructions to transfer control to the computer. Without the explicit instructions, the EG&G lock-in amplifiers will try to maintain control of the interface bus and lock out other communications. The other routines listed below perform various other functions to maintain reliable communication between the computer and remote device as described in the subroutine headings.

```
#include    "C:\lw\include\lwsystem.h"
#include    "C:\lw\include\gpib.h"

/*These routines perform the low level I/O to all devices through the GPIB (IEEE) interface. */
/* All device calls are performed through these routines*/

#define TALK 64
#define LIA_TIME 12
#define LISTEN 32
```

```
#define GPIB0_ADDRESS 0  /*GPIB Board address*/
static int brd_addr = -1;      /*use it to hold board address*/
static int board = -1;         /*used to hold access code for board calls */

/*********************************************************************/
/*GEN_init : Initialize device and make sure it will respond to GPIB board
/*     It also sets GPIB board in computer to controller state
/* PARAMETERS:
/*     addr: IEEE address of device (0-31)
/*          addr=0 corresponds to just resetting the board and can be used
/*          to clear some IO errors.  If addr>31, routines use addr%32
/*RETURN:
/*     -1 on error, 0 if successful
/*********************************************************************/
int GEN_init(int addr)
{
        int n;
        char temp[6];

        board = ibfind("GPIB0");
        brd_addr = GPIB0_ADDRESS;
        if (board < 0) {
           printf("ERROR: GEN_init(%i)\n",addr);
           return -1;
        }
        ibsic(board); /*clear interface*/
        ibcac(board,1); /*make board controller */
        if (ibsta < 0)
          printf("ERROR: GEN_init(%i) -board 0 not in command\n",addr);
        ibsic(board); /* repeat interface clear (it helps in some cases)*/
        if (ibsta < 0)
          printf("ERROR: GEN_init(%i) -board 0 not in command\n",addr);
        if (addr%32 > 0)  {
          temp[0] = 95;    /* untalk */
          temp[1] = '?';
          temp[2] = addr % 32 + LISTEN;
          temp[3] = 0;
          ibcmd(board,temp,3);
          temp[2] = 0;
          ibcmd(board,temp,2);
          if (ibsta < 0)
            printf("init Command not sent correctly:%x %i %i addr=%i,board=%i\n"
                  ,ibsta,iberr,ibcnt,addr,brd_addr);
          ibsre(board,1);
        }
        return 0;
}

/*********************************************************************/
/*GEN_write: performs low-level write to device
/*     This routine address the device, sends a string, and
/*     unaddresses the device
/*PARAMETERS:
```

```
/*     addr: IEEE address of the device
/*     cmd : string to send to device
/*     len : number of characters in the string to send to the device
/*RETURN:
/*     -1 on error, otherwise number of characters sent
/*********************************************************************/
int GEN_write(int addr, char *cmd, int len)
{
        int m, n;
        char temp[5];
        char buff[110];

        temp[0] = 95;   /* untalk */
        temp[1] = '?';
        temp[2] = addr % 32 + LISTEN;
        temp[3] = 0;
        n = -1;
        for (m = 0; m < 5 && n < 0; m++) {    /*try up to 5 times to access the device*/
          ibcmd(board,temp,3);
          if ((n = ibsta) >= 0)
            ibsre(board,1);
          else {
            printf("ERROR: GEN_write(%i,%s,%i) cannot access device\n",addr,cmd,len);
            GEN_init(0);
          }
        }

        temp[2] = addr % 32 + LISTEN;
        temp[3] = brd_addr + TALK;
        temp[4] = 0;
        ibcmd (board, temp, 4);
        if (ibsta < 0)
          printf("ERROR: GEN_write(%i,%s,%i) string not sent\n",addr,cmd,len);
        ibwrt(board,cmd,len);
        n = ibcnt;
        temp[2] = 0;
        ibcmd(board,temp,2);
        return n;
}

/*********************************************************************/
/*GEN_write_file: performs low-level write to device from a file
/*     This routine address the device, sends the contents of a file, and
/*     unaddresses the device
/*     SEE ALSE GEN_write()
/*PARAMETERS:
/*     addr: IEEE address of the device
/*     cmd : name of file to send
/*RETURN:
/*     IEEE status byte (<0 on error)
/*********************************************************************/
int GEN_write_file(int addr, char *cmd)
{
```

```
        int m, n;
        char temp[5];
        temp[0] = 95;   /* untalk */
        temp[1] = '?';
        temp[2] = addr % 32 + LISTEN;
        temp[3] = 0;
        n = -1;
        for (m = 0; m < 5 && n < 0; m++) {  /*make up to 5 attempts to communicate to device*/
          ibcmd(board,temp,3);
          if (ibsta >= 0)
            ibsre(board,1);
          n = ibsta;
          if (n < 0) {
            printf("ERROR: GEN_write_file(%i,%s) cannot access device\n",addr,cmd);
            GEN_init(0);
          }
        }

        temp[2] = addr % 32 + LISTEN;
        temp[3] = brd_addr + TALK;
        temp[4] = 0;
        ibcmd (board, temp, 4);
        printf ("\nsending file %s: stat=%i, err=%i",cmd,ibsta,iberr);
        ibwrtf(board,cmd);  /*write file*/
        printf ("\nfile %s sent: stat=%i, err=%i, count=%i",cmd,ibsta,iberr,ibcnt);
        temp[2] = 0;
        ibcmd (board, temp,2);
        return ibsta;
}

/*********************************************************************/
/*GEN_Read: performs low-level read from device
/*     This routine address the device, receives a string, and
/*     unaddresses the device
/*PARAMETERS:
/*     addr: IEEE address of the device
/*     cmd : string buffer to hold string
/*     len : maximum number of characters to read from device
/*           NOTE: specifiying len too large is not a problem.  It will
/*                 stop reading when device sends EOI signal.  Just be
/*                 sure cmd is large enough to hold at least len characters
/*RETURN:
/*     -1 on error, otherwise number of characters read
/*********************************************************************/
int GEN_read(int addr, char *cmd, int len)
{
        int n, m;
        char temp[5];
        int srq;

        temp[0] = 95;    /* untalk */
        temp[1] = '?';
        temp[2] = addr % 32 + LISTEN;
```

```
        temp[3] = 0;
        n = -1;
        for (m = 0; m < 5 && n < 0; m++) {  /* make up to 5 attempts to communicate to device */
          ibcmd(board,temp,3);
          if (ibsta >= 0)
            ibsre(board,1);
          n = ibsta;
          if (n < 0)
            GEN_init(0);
        }
        temp[2] = addr % 32 + TALK;
        temp[3] = brd_addr + LISTEN;
        temp[4] = 0;
        ibcmd (board, temp, 4);
        if (ibsta < 0)
          printf("ERROR: GEN_write(%i,(buffer),%i) cannot access device\n",addr,len);
        ibrd(board,cmd,len);
        n = ibcnt;
        temp[2] = 0;
        ibcmd(board,temp,2);
        return n;
}

/*****************************************************************/
/*GEN_rsp: performs serial poll on device
/*    This routine addresses the device, gets service request byte, and
/*    unaddresses the device. See GPIB manual for more information on serial polling
/*PARAMETERS:
/*    addr: IEEE address of the device
/*RETURN:
/*    -1 on error, otherwise service request register (SRQ byte). The
/*    meaning of this byte is device dependent-see appropriate manuel
/*****************************************************************/
int GEN_rsp(int addr)
{
        char cmd[6];
        char end;
        char srq;
        int n, m;

        cmd[0] = 95;   /* untalk */
        cmd[1] = '?';  /* unaddress */
        cmd[2] = addr % 32 + LISTEN;
        cmd[3] = 0;
        n = -1;
        for (m = 0; m < 5 && n < 0; m++) {
          ibcmd(board,cmd,3);
          if (ibsta >= 0)
            ibsre(board,1);
          n = ibsta;
          if (n < 0)
            GEN_init(0);
        }
```

```
        cmd[4] = 24;    /* Serial Poll Enable */
        cmd[2] = TALK + addr % 32;
        cmd[3] = brd_addr + LISTEN;

        ibcmd (board, cmd, 5);
        ibrd(board,&srq,1);
        cmd[2] = 25;    /*Serial Poll Disable */
        ibcmd(board,cmd,3);
        return srq;
}

/*********************************************************************/
/*GEN_trigger: sends GET (group execute trigger) signal to a device
/*      This routine address the device, sends GET signal, and
/*      unaddresses the device. See GPIB manual for more information on GET
/*PARAMETERS:
/*      addr: IEEE address of the device
/*RETURN:
/*      GPIB status byte (<0 on error)
/*********************************************************************/
int GEN_trigger(int addr)
{
        char cmd[5];
        int n, m;

        cmd[0] = 95;    /* untalk */
        cmd[1] = '?';   /* unaddress */
        cmd[2] = addr % 32 + LISTEN;
        cmd[3] = 0;
        n = -1;
        for (m = 0; m < 5 && n < 0; m++) {   /*make 5 attempts if the interface is stuck*/
          ibcmd(board,cmd,3);
          if (ibsta >= 0)
            ibsre(board,1);
          n = ibsta;
          if (n < 0)
            GEN_init(0);
        }

        cmd[3] = 8;     /* GET command (trigger)*/
        cmd[4] = 0;
        ibcmd (board, cmd, 4);
        cmd[2] = 0;
        ibcmd (board,cmd,2);
        return ibsta;
}

/*********************************************************************/
/*GEN_clr: sends clear signal to a device
/*      This routine address the device, sends signal, and
/*      unaddresses the device. See GPIB manual for more information on IBCLR
/*PARAMETERS:
```

```
/*     addr: IEEE address of the device
/*RETURN:
/*     GPIB status byte (<0 on error)
/*********************************************************************/
int GEN_clr(int addr)
{
        char cmd[4];

        cmd[0] = 95;   /* untalk */
        cmd[1] = '?';    /* unaddress */
        cmd[2] = addr % 32 + LISTEN;
        cmd[3] = 0;
        ibcmd(board,cmd,3);
        ibsre(board,0);
        cmd[0] = 4;  /* device clear */
        cmd[1] = 0;
        ibcmd(board,cmd,1);
        cmd[0] = 95;   /* untalk */
        cmd[1] = '?';    /* unaddress */
        cmd[2] = 0;
        ibcmd(board,cmd,1);
        return ibsta;
}

/*********************************************************************/
/*GEN_loc: sends go to local signal to a device
/*     This routine address the device, sends go to local signal, and
/*     unaddresses the device.  See GPIB manual for more information on IBLOC
/*PARAMETERS:
/*     addr: IEEE address of the device
/*RETURN:
/*     GPIB status byte (<0 on error)
/*********************************************************************/
int GEN_loc(int addr)
{
        char cmd[4];

        cmd[0] = 95;   /* untalk */
        cmd[1] = '?';    /* unaddress */
        cmd[2] = addr % 32 + LISTEN;
        cmd[3] = 0;
        ibcmd(board,cmd,3);
        ibsre(board,0);
        cmd[0] = 95;
        cmd[1] = '?';    /* unaddress */
        cmd[2] = 0;
        ibcmd(board,cmd,1);
        return ibsta;
}

/*********************************************************************/
/*GEN_remote: sends SRE (set remote enable) signal to a device
/*     This routine address the device, sends go to remote signal, and
```

```
/*     unaddresses the device.  See GPIB manual for more information on IBSRE
/*PARAMETERS:
/*     addr: IEEE address of the device
/*RETURN:
/*     GPIB status byte (<0 on error)
/*****************************************************************/
int GEN_remote(int addr)
{
        char cmd[4];

        cmd[0] = 95;   /* untalk */
        cmd[1] = '?';    /* unaddress */
        cmd[2] = 0;
        ibcmd(board,cmd,2);
        ibsre(board,1);
        cmd[1] = addr % 32 + LISTEN;
        cmd[2] = 0;
        ibcmd(board,cmd,1);
        cmd[0] = 95;
        cmd[1] = '?';    /* unaddress */
        cmd[2] = 0;
        ibcmd(board,cmd,1);
        return ibsta;
}

/*****************************************************************/
/*GEN_timeout: sets timeout limit for GPIB calls.
/*     NOTE: This is a board command and affects all IO performed to any
/*           device!!!
/*     The timeout limit is just how long the board will wait for any opperation
/*     to finish before aborting
/*PARAMETERS:
/*     tmo : timeout limit code - see manual under IBTMO for the values used
/*RETURN:
/*     GPIB status byte (<0 on error)
/*****************************************************************/
int GEN_timeout(int tmo)
{
        ibtmo(board,tmo);   /*set timeout limit for all board functions */
        return ibsta;
}

/*****************************************************************/
/*GEN_unaddress: sends unaddress signal to a device
/*     This routine address the device, sends unaddress signal, and
/*     unaddresses the device.  This routine is rarely used because the
/*     unaddress command is usually sent at the end of each service routine.
/*PARAMETERS:
/*     addr: IEEE address of the device
/*RETURN:
/*     GPIB status byte (<0 on error)
/*****************************************************************/
```

int GEN_unaddress (int addr)

```
{
        char cmd[3];

        cmd[0] = 95;    /* untalk */
        cmd[1] = '?';   /* unaddress */
        cmd[2] = 0;
        ibcmd(board,cmd,2);
        return ibsta;
}
```

Appendix B : Microwave Measurement Control

The control program used for the microwave measurements, referred to as "Q_find," is less generalized than the program Mscan used for dc and low frequency measurements. The primary measurement is to measure the Q of several different resonant modes in a microwave cavity as a function of temperature and magnetic field. For a complete description of the microwave measurement technique, see reference 1. The only other task for Q_find to achieve is to find the modes in the resonator. Many of the basic features of Q_find are the same as Mscan in terms of controlling the interface, maintaining temperature control, and controlling the magnet. However, the primary measurement routines are considerably different.

Searching for the modes is performed by the main function of Q_find, which increments the frequency of the applied microwave signal and checking to see if a large signal is measured by the oscilloscope. When a large signal is measured, this is an indication of a resonance mode of the resonator so Q_find will either stop or record the frequency of the mode and continue looking for the next mode. The primary data is taken in the subroutine "Scan." In this routine, the frequency is set to a value slightly below the center frequency of the resonant mode of the microwave cavity. The signal level is then measured and the frequency is incremented by a small value. In this way, a set of microwave amplitude versus frequency data can be taken to map the frequency spectrum of the resonant mode.

Partial Listing of Q_find.c:

```
#include <dos.h>
#include <sys\types.h>
#include <sys\timeb.h>
#include <time.h>
#include <graph.h>
#include <color.h>
#include <bios.h>
#include <stdio.h>
#include <math.h>
#include "C:\lw\include\osc_def.h"
#include "C:\lw\include\lwsystem.h"
#include "C:\lw\include\gpib.h"
```

```
#define T_GAIN -5
#define T_RATE -4
#define T_RESET -3
#define T_POWER -1
#define T_SETPOINT -2
#define T_CONTROL 0
#define T_SAMPLE 1

#define SMOOTH .20
#define HOME 0x4700
#define MAX_POINTS 100
#define TEMP_ADDR 10
#define MODEL70_ADDR 30
#define MAX_MODE 100
#define LOG_SQR_2 .34657359
#define START_A 1000
#define TIMEOUT 100
#define A_ERROR .001
#define FAST_COUNT 20
#define SLOW_COUNT 50
#define FAST_POINTS 128
#define SLOW_POINTS 512
#define SCOPE_ADDR 14
#define OSC_ADDR 15
#define ENTER 13
#define TRUE !0
#define MIN_STEP 1e3
#define CHAN_MAX 10          /*number of reads to maximize channel */
#define MAX_INPUT 120
#define BLOCK1 3
#define BLOCK2 8
#define BLOCK3 11
#define BLOCK4 21
#define BLOCK5 23

enum PRINTS{ENVELOPE, RESOLUTION, COMMAND, C_FREQUENCY, CHECK, EDGE,
        RANGE, DELAY, AUTO, TITLE, TRANSFER, NOISE, WIDTH, LEVEL, MODES,
        INCREMENT, MAX_PRINT
};
char cursor[4]={ '|' , '/' , '-' , '\\' };
double data[3][600]; /* buffer for use by all subroutines */

main(){
    char f_name[40]; FILE *par_file;
    struct find_t r_path;  int auto_stop;  char stop_file[100];  FILE *sfile; int list_modes;  double a;
    char trans_name[80]; char input[MAX_INPUT];  double value,value1,value2,value3,q_error;
    char code, drive[4], direct[80], char text2[160],char text3[30], int print[MAX_PRINT];

    struct mod {  /** the primary structure used to set values on each mode*/
        double freq;  /**mode center frequency*/
        float power;  /*power level to use in measuring the mode*/
        float q;      /*the frequency width to use for measuring the mode*/
        float delay;  /*time position to measure the oscillator level */
        int scan_res;  /*number of frequency points to measure*/
```

```
        int envelope;  /*length of time to measure each point*/
        } mode[MAX_MODE+1];
 int loc,cur_mode;   int do_noise,old_mode;  float q_val[MAX_MODE];  int scope_res; unsigned m;
 float cur_width; double cur_power; double noise;  int cur_scan_res;   int n,b;  int input_ready;
double edge;  long check_time;  long check_int;  long l_delay;  char text[200]; long time0;int paused;
 int measure_q, find_q; double c; float delta_t; int num_steps;
 double fun,cur,cur_freq,freq_step,high_freq,low_freq,half,half_freq,max,range,small_step,max_freq;
 noise=.138;
 check_int=5000;
 freq_step=1e-5;
 *trans_name=*f_name=*direct=0;
 range=1e-4;
 edge=1e-6;
 cur_mode=0;
 do_noise=!TRUE;
 for(n=0;n<MAX_MODE;n++){
        mode[n].power=-110;
        mode[n].freq=-1;
        mode[n].delay=mode[n].q=0;
        q_val[n]=-1;
 }
 if((par_file = fopen ("c:\\lw\\programs\\modes.par", "r+b"))!=NULL){
        fread ((char *)mode,sizeof mode,1,par_file);
        fread ((char *)&noise,sizeof(double),1,par_file);
        fread ((char *)&check_int,sizeof(long),1,par_file);
        fread ((char *)&freq_step,sizeof(double),1,par_file);
        fread ((char *)&range,sizeof(double),1,par_file);
        fread ((char *)&edge,sizeof(double),1,par_file);
        fread ((char *)&cur_mode,sizeof(int),1,par_file);
        fread (f_name,30,1,par_file);
        fread (trans_name,80,1,par_file);
        fread (direct,80,1,par_file);
        fclose (par_file);
 }
 else
        beep();
 for(n=0;n<MAX_PRINT;n++)
        print[n]=1;
 for(n=0;n<MAX_MODE;n++){
        if (mode[n].power<-110) mode[n].power=-110;
        if(mode[n].power>20) mode[n].power=20;
        if(mode[n].freq==-1.0) mode[n].freq=-1001.0;
 }
 init_time();
 loc=0;
 input_ready=!TRUE;
 auto_stop=!TRUE;
 measure_q=!TRUE;
 find_q=!TRUE;
 check_time=-check_int;
 paused=TRUE;
 cur_freq=fabs(mode[cur_mode].freq);
 cur_power=mode[cur_mode].power;
 cur_width=mode[cur_mode].q;
```

```
cur_scan_res=mode[cur_mode].scan_res;
scope_res=mode[cur_mode].envelope;
input[0]=0;
  delta_t=mode[cur_mode].delay;
num_steps=0;
q_error=.1;
l_delay=100; /* ms */
n=SCOPE_init(SCOPE_ADDR);
OSC_init(OSC_ADDR);
TEMP_init(TEMP_ADDR);
/* find time of falling edge */
SCOPE_setup();
*stop_file=0;
if(_setvideomode(_TEXTC80)==0){
      printf("error in mode");
      _bios_keybrd(_KEYBRD_READ);
}
_clearscreen(_GCLEARSCREEN);
_displaycursor(_GCURSOROFF);
_settextcolor(RED+BLINK);
_settextposition(1,1);
_outtext("PAUSED");
_settextposition(BLOCK5,1);
_settextcolor(BLACK);
_outtext("                                                    ");
while(input[0]!='q'&& input[0]!='Q'){
     if(print[RESOLUTION]){
                          _settextcolor(LIGHTMAGENTA);
                          _settextposition(BLOCK3+1,1);
                          sprintf(text,"Resolution   %.3i points",cur_scan_res);
                          _outtext(text);
     }
     if(print[COMMAND]) {
                          _settextcolor(LIGHTRED);
                          _settextposition(BLOCK3+8,1);
                          _outtext("Go to magnet control          <Home>: clear scope");
                          _settextposition(BLOCK3+9,1);
                          _outtext("Kill mode (or restore)        View modes");
                          _settextposition(BLOCK3+2,1);
                          _outtext("Begin Scan");
     }
     if(print[ENVELOPE]){
                          _settextcolor(LIGHTMAGENTA);
                          _settextposition(BLOCK3+3,1);
                          sprintf(text,"Envelope   %.3i points",scope_res);
                          _outtext(text);
     }
     if(print[CHECK])  {
                     _settextcolor(LIGHTRED);
                     _settextposition(BLOCK2+1,1);
                     sprintf(text,"Check time : %.3f   ",(float)check_int/1000);
                     _outtext(text);
     }
     if(print[EDGE]) {
```

```
                    _settextcolor(LIGHTRED);
                    _settextposition(BLOCK3+5,1);
                    sprintf(text,"0-time  : %.6e sec   ",edge);
                    _outtext(text);
}
if(print[RANGE]) {
                    _settextcolor(LIGHTRED);
                    _settextposition(BLOCK3+6,1);
                    sprintf(text,"Zap time: %.6e sec   ",range);
                    _outtext(text);
}
if(print[TITLE]) {
                _settextcolor(LIGHTRED);
                _settextposition(BLOCK3,1);
                sprintf(text2,"%s%s.*",direct,f_name);
                sprintf(text,"Title: %-28.28s",text2);
                _outtext(text);
}
if(print[AUTO]){
          _settextcolor(LIGHTRED);
          _settextposition(BLOCK2,1);
          if(auto_stop){
               sprintf(text,"Auto stop %s",stop_file);
               _outtext(text);
          }
          else
               _outtext("Auto stop OFF       ");
}
if(print[TRANSFER]){
                    _settextcolor(LIGHTRED);
                    _settextposition(BLOCK3+7,1);
                    sprintf(text,"Xpell mode   ");
                    /*sprintf(text,"X_transfer: %s",trans_name);*/
                    _outtext(text);
}
if(print[NOISE]){
                _settextcolor(LIGHTRED);
                _settextposition(BLOCK2+2,1);
                sprintf(text,"Noise level:%f  V  ",noise);
                _outtext(text);
}
if(print[WIDTH]){
                    _settextcolor(LIGHTMAGENTA);
                    _settextposition(BLOCK1+3,1);
                    sprintf(text,"Width    : %g MHz  ",cur_width*1000);
                    if(do_noise)
                         _outtext("Width    : NONE");
                    else
                         _outtext(text);
}
if(print[LEVEL]){
                _settextcolor(LIGHTMAGENTA);
                _settextposition(BLOCK1+2,1);
                sprintf(text,"Level    : %.2g dBm",cur_power);
```

```
                _outtext(text);
    }
    if(print[DELAY]) {
                    _settextcolor(LIGHTMAGENTA);
                    _settextposition(BLOCK3+4,1);
                    sprintf(text,"Delay   : %.6e u sec",delta_t*1e6);
                    _outtext(text);
    }
    if(print[MODES]){
        _settextposition(1,32);
        _settextcolor(CYAN);
        _outtext("MODES:  GHz       dbm  width  delay");
        for(m=n=0;n<MAX_MODE;n++)
            if(fabs(mode[n].freq)>.01 && fabs(mode[n].freq)<MAX_FREQUENCY){
                _settextposition(BLOCK3+m,34);
                if(q_val[n]>0)
                    sprintf(text,"%11.7f q=%.6E d=%.6E",mode[n].freq,q_val[n]
                            ,6.283185307*mode[n].freq*1e9/q_val[n]);
                else
                    sprintf(text,"%11.7f q=ERROR",mode[n].freq);
                _outtext(text);
                _settextposition(2+m++,36);
                sprintf(text,"M%i %13.9f %7.2f %8.4f
                            %5.3f",n,fabs(mode[n].freq),mode[n].power
                            ,mode[n].q*1000,mode[n].delay*1e6);
                _outtext(text);
                if(n==cur_mode){
                    _settextposition(1+m,36);
                    _settextcolor(LIGHTCYAN);
                    sprintf(text,"M%i",n);
                    _outtext(text);
                    _settextcolor(CYAN);
                }
                if(mode[n].freq<0){
                    _settextposition(1+m,36);
                    _settextcolor(BLUE);
                    _outtext("M");
                    _settextcolor(CYAN);
                }

            }
        if(m<MAX_MODE){
            _settextposition(2+m,36);
            _outtext("                              ");
            _settextposition(BLOCK3+m,34);
            _outtext("                               ");
        }
        print[MODES]=!TRUE;
}
if(print[C_FREQUENCY]){
    _settextcolor(LIGHTMAGENTA);
    _settextposition(BLOCK1,1);
    sprintf(text,"Frequency: %.9f",cur_freq);
    _outtext(text);
```

```
}
if(print[INCREMENT]){
      _settextcolor(LIGHTRED);
      _settextposition(BLOCK1+1,1);
      sprintf(text,"Increment: %g MHz",freq_step*1000);
      _outtext(text);
}
for(m=0;m<MAX_PRINT;m++)
     print[m]=!TRUE;
if(cur_freq<MAX_FREQUENCY && cur_freq>.01 && !paused &&
                (n=OSC_set_freq(cur_freq))<0)
    printf("error: ibsta=%x,iberr=%x\n",n,iberr);
time0=s_time();
while(time0+1_delay>s_time()){
           _settextposition(BLOCK5,loc+1);
           text[0]=cursor[(s_time()%400)/100];
           text[1]=0;
           _settextcolor(LIGHTMAGENTA);
           _outtext(text);
           _settextcolor(BLACK);
  if(_bios_keybrd(_KEYBRD_READY)!=0){
       input[loc]= (m=_bios_keybrd(_KEYBRD_READ))%256;
       if((m==0xE08 || m==0x4B00)&&loc>0){
          input[--loc]=0;
          _settextcolor(BLACK);
          _settextposition(BLOCK5,1+loc);
          _outtext("  ");
       }
       if(m==HOME){
          SCOPE_init(SCOPE_ADDR);
          OSC_set_power(cur_power);
          pause(1);
          SCOPE_set_range(edge-range/10,range);
          pause(1);
          /*SCOPE_close();*/
          /*OSC_set_power(cur_power);*/
       }
       if(input[0]=='P' || input[0]=='p'){
          paused=!paused;
          input[0]=0;
          _settextcolor(RED+BLINK);
          _settextposition(1,1);
          if(paused){
               _outtext("PAUSED");
               /* OSC_close();  do not set to local */
          }
          else{
               _outtext("      ");
               check_time=s_time()+check_int;
               OSC_init(OSC_ADDR);
               OSC_set_power(cur_power);
          }
          _settextcolor(BLACK);
      }
```

```
    if((input[loc]==ENTER && loc!=0) || loc==MAX_INPUT-1){
        input[loc]=0;
        loc=0;
        input_ready=TRUE;
        _settextcolor(BLACK);
        _settextposition(BLOCK5,1);
        _outtext("
    }
    else if(loc<MAX_INPUT-1 && input[loc]>=' '){
        _settextposition(BLOCK5,loc+1);
        text[0]=input[loc++];
        text[1]=0;
        _settextcolor(GREEN);
        _outtext(text);
        _settextcolor(BLACK);
    }
    else
        input[loc]=0;
}
else if(input_ready){
  value2=value1=value3=-1e7;
  code=0;
  sscanf(input,"%c%lf%lf%lf",&code,&value1,&value2,&value3);
  /*printf("read: code= '%c' value1=%f  value2=%f\n",code,value1,value2);*/
  switch(code){
      case  0 :
            break;
      case '-':
            if(value1<1e-6)
                freq_step*=-1;
            else
                freq_step=-value1;
            print[INCREMENT]=TRUE;
            break;
      case 'c':
      case 'C':
            if(value1>=0)
                check_int=(int)(value1*1000);
                print[CHECK]=TRUE;
            break;
      case 'a':
      case 'A':
            auto_stop=!auto_stop;
            for(n=1;input[n]==' ';n++);
            if (input[n]!=0)
                sscanf(input+1,"%s",stop_file);
            print[AUTO]=TRUE;
            break;
      case 'E':
      case 'e':
            if(value1>0){
                scope_res=value1;
                print[ENVELOPE]=TRUE;
            }
```

```
                break;
            case 'r':
            case 'R':
                if(value1>0){
                    cur_scan_res=value1;
                    if(cur_scan_res>MAX_POINTS)
                        cur_scan_res=MAX_POINTS;
                    print[RESOLUTION]=TRUE;
                }
                break;
            case 'd':
            case 'D':
                if(value1>0){
                    delta_t=value1/1e6;
                    print[DELAY]=TRUE;
                }
                break;
            case 't':
            case 'T':
                for(n=1;input[n]==' ';n++);
                if (input[n]!=0){
                    _splitpath(input+n,drive,text,f_name,text2);
                    if(*text!=0)
                        strcpy(direct,text);
                }
                print[TITLE]=TRUE;
                break;
            case 'x':
            case 'X':
              /*    if(input[1]==0 || input[2]==0 && *f_name!=0){
                    sprintf(text,"%s%s.*",direct,f_name);
                    sprintf(text2,"%s%s.*",trans_name,f_name);
                    transfer(text,text2);
                }
                else{
                    sscanf(input+1,"%s",trans_name);
                    print[TRANSFER]=TRUE;
                }   */
                print[MODES]=TRUE;
                if((int)value1>=0 && (int)value1<MAX_MODE)
                    if(mode[(int)value1].freq>-1000)
                        mode[(int)value1].freq*=-1;
                break;
            case 'b':
            case 'B':
                if(do_noise){
                    do_noise=!TRUE;
                    _settextposition(BLOCK5+1,1);
                    _outtext("                                ");
                    cur_mode=old_mode;
                }
par_file = fopen ("c:\\lw\\programs\\modes.par", "w+b");
fwrite ((char *)mode,sizeof mode,1,par_file);
    fwrite ((char *)&noise,sizeof(double),1,par_file);
```

```
    fwrite ((char *)&check_int,sizeof(long),1,par_file);
    fwrite ((char *)&freq_step,sizeof(double),1,par_file);
    fwrite ((char *)&range,sizeof(double),1,par_file);
    fwrite ((char *)&edge,sizeof(double),1,par_file);
    fwrite ((char *)&cur_mode,sizeof(int),1,par_file);
    fwrite (f_name,30,1,par_file);
    fwrite (trans_name,80,1,par_file);
    fwrite (direct,80,1,par_file);
fclose (par_file);
                sprintf(text,"%s%s.*",direct,f_name);
                n=0;
                while(n>-1 && (b=_dos_findfirst(text,_A_NORMAL|_A_RDONLY
                                        |_A_ARCH,&r_path))==0){
                        if(f_name[7]>='0' && f_name[7]<'9')
                            f_name[7]++;
                        else if(f_name[7]=='9' && f_name[6]>='0' && f_name[6]<'9'){
                            f_name[6]++;
                            f_name[7]='0';
                        }
                        else{
                                        _settextposition(BLOCK5,1);
                                        _settextcolor(YELLOW+BLINK);
                                        sprintf(text,"  Files already exist");
                                        _outtext(text);
                                        beep();
                                        pause(5);
                                        n=-1;
                        }
                        if(n>-1)
                            sprintf(text,"%s%s.*",direct,f_name);
                }
                _settextposition(BLOCK5,1);
                _settextcolor(YELLOW);
                _outtext(text);
                while(_bios_keybrd(_KEYBRD_READY))
                    if(_bios_keybrd(_KEYBRD_READ)==0x11b)
                        b=0;
                if(b!=0){
                    Scan(mode,direct,f_name,edge,range,cur_mode,q_val,SMOOTH,noise);
                                /* 3 is the smoothing level */
                    if(_setvideomode(_TEXTC80)==0){
                        printf("error in mode");
                        _bios_keybrd(_KEYBRD_READ);
                    }
                }
                _settextposition(BLOCK5,1);
                _settextcolor(BLACK);
                _outtext("                    ");
                for(b=0;b<MAX_PRINT;b++)
                    print[b]=TRUE;
                break;
            case 'v':
            case 'V':
                for(n=1;input[n]==' ';n++);
```

```
                *text=*text2=0;
                if (input[n]!=0)
                    _splitpath(input+n,drive,text,text2,text3);
                if(*text==0)
                    strcpy(text,direct);
                if(*text2==0)
                    strcpy(text2,f_name);
                Display(text,text2,SMOOTH);
                _settextposition(BLOCK5,1);
                _settextcolor(BLACK);
                _outtext("                ");
                for(b=0;b<MAX_PRINT;b++)
                    print[b]=TRUE;
                break;
            case 'g':
            case 'G':
                MAGNET_control();
                for(n=0;n<MAX_PRINT;n++)
                    print[n]=TRUE;
                _settextcolor(BLACK);
                _clearscreen(_GCLEARSCREEN);
                _displaycursor(_GCURSOROFF);
                if(paused){
                    _settextcolor(RED+BLINK);
                    _settextposition(1,1);
                    _outtext("PAUSED");
                    _settextcolor(BLACK);
                }
                break;
            case 'Z':
            case 'z':
                if(value1>=0){
                    range=value1;
                    print[RANGE]=TRUE;
                    break;
                }
            case 'o':
            case 'O':
            case '0':
                if(value1>=0){
                    edge=value1;
                    print[EDGE]=TRUE;
                    break;
                }
SCOPE_init(SCOPE_ADDR);
OSC_set_power(cur_power);
SCOPE_digitize(3,64,512);      /*read input pulse(512 points,envelope of 64)*
 time0=s_time();
 SCOPE_ready(time0,300);
 n=SCOPE_read();
 edge=SCOPE_find_edge(n,2);
 range=data[0][n-1]-data[0][0];
 SCOPE_c_height(edge);  /*mark edge where measurements will begin*/
                print[EDGE]=TRUE;
```

```
        print[RANGE]=TRUE;
        /*SCOPE_close();*/
        break;
 case 'h':
 case 'H':
        SCOPE_init(SCOPE_ADDR);
        OSC_set_power(cur_power);
        value2=SCOPE_f_height(edge,range/20,(int)value1);
        if(SCOPE_set_range(edge,edge/900)<0)
              printf("range error: %x",ibsta);
        value=SCOPE_height(scope_res);
        SCOPE_set_range(edge-range/10,range);
        sprintf(text,"Height= %g\nMax height=%g",value,value2);
        _settextposition(BLOCK4,1);
        _settextcolor(YELLOW);
        _outtext(text);
        /*SCOPE_close();*/
        break;
case 'i':
case 'I': if(fabs(value1)<20){
              freq_step=value1/1000;
              print[INCREMENT]=TRUE;
        }
        break;
case 'n':
case 'N':
        if(do_noise){
                    noise=0.0;
                    if(cur_scan_res>0){
                          SCOPE_init(SCOPE_ADDR);
                          OSC_set_power(cur_power);
                          if(SCOPE_set_range(edge+delta_t,(edge+delta_t*2)/900)<0)
                               printf("range error: %x",ibsta);
                          _settextcolor(YELLOW);
                          for(b=0;b<cur_scan_res;b++){
                               noise+=(value1=SCOPE_height(scope_res));
                               sprintf(text," noise=%f                ",value1);
                               _settextposition(BLOCK5+1,1);
                               _outtext(text);
                          }
                          SCOPE_set_range(edge-range/10,range);
                          noise/=cur_scan_res;
                          /*SCOPE_close();*/
                    }
                    mode[MAX_MODE].freq=cur_freq;
                    mode[MAX_MODE].power=cur_power;
                    mode[MAX_MODE].q=cur_width;
                    mode[MAX_MODE].delay=delta_t;
                    mode[MAX_MODE].scan_res=cur_scan_res;
                    mode[MAX_MODE].envelope=scope_res;
                    value1=old_mode;
                    print[NOISE]=TRUE;
      } /* then continue with M */
      else if(value1>=0){
```

```
                noise=value1;
                print[NOISE]=TRUE;
                break;
            }
            else{
                old_mode=cur_mode;
                do_noise=TRUE; /* set parameters for noise */
                _settextcolor(YELLOW);
                _settextposition(BLOCK5+1,1);
                _outtext("Set up noise check-press n to begin");
                value1=MAX_MODE;
            }               /* continue on with M100 */
    case 'm':
    case 'M': n=(int)fabs(value1);
            if(!do_noise && n>=MAX_MODE)
                n=MAX_MODE+100;
            if(do_noise && n!=MAX_MODE){
                do_noise=!TRUE;
                _settextposition(BLOCK5+1,1);
                _outtext("                              ");
            }
            if(n<MAX_MODE+1 && fabs(mode[n].freq)>.01 && fabs(mode[n].freq<20)){
                print[MODES]=TRUE;
                cur_freq=fabs(mode[cur_mode=n].freq);
                cur_power=mode[cur_mode].power;
                cur_width=mode[cur_mode].q;
                delta_t=mode[cur_mode].delay;
                scope_res=mode[cur_mode].envelope;
                cur_scan_res=mode[cur_mode].scan_res;
                print[LEVEL]=print[WIDTH]=print[C_FREQUENCY]=print
                            [INCREMENT]=print[DELAY] =print[RESOLUTION]
                            =print[ENVELOPE]=TRUE;
            }
            OSC_init(OSC_ADDR);
            OSC_set_power(cur_power);
            OSC_set_freq(cur_freq);
            /*OSC_close();*/
            break;
    case 'w':
    case 'W':
            if(value1>0){
                cur_width=value1/1000;
                print[WIDTH]=TRUE;
            }

            break;
    case 'l':
    case 'L':
            OSC_init(OSC_ADDR);
            OSC_set_power(cur_power);
            if(value1<-1e6);
                /*  cur_power=OSC_read_power(); */
            else if(value1>=-20 && value1<MAX_AMPLITUDE)
                OSC_set_power(cur_power=value1);
```

```
                /*OSC_close();*/
                print[LEVEL]=TRUE;
                break;
            case 'f':
            case 'F':
                OSC_init(OSC_ADDR);
                OSC_set_power(cur_power);
                if(value1<0){
                    cur_freq=OSC_read_freq();
                    /*if(paused)
                        OSC_close();*/
                }
                else  {
                    cur_freq=value1;
                    OSC_set_freq(cur_freq);
                }
                print[C_FREQUENCY]=TRUE;
                break;
            case 'k':
            case 'K':
                print[MODES]=TRUE;
                if((int)value1>=0 && (int)value1<MAX_MODE){
                    if(mode[(int)value1].freq<-1000)
                        mode[(int)value1].freq+=2000;
                    else
                        mode[(int)value1].freq-=2000;
                }
                break;
            case 's':
            case 'S':
                print[MODES]=TRUE;
                if(value2<0 && value1<0)
                    n=cur_mode;
                else if((n=(int)fabs(value1))>MAX_MODE-1)
                    n=MAX_MODE-1;
                cur_mode=n;
                mode[n].freq=(value2>-1e6)?value2:cur_freq;
                mode[n].power=(value3>-1e6)?value3:cur_power;
                mode[n].q=cur_width;
                mode[n].delay=delta_t;
                mode[n].scan_res=cur_scan_res;
                mode[n].envelope=scope_res;
                break;
            default:  sscanf(input,"%lf",&freq_step);
                freq_step/=1000;
                print[INCREMENT]=TRUE;
                break;
        }
        if(*input!='q' && *input!='Q')
            *input=0;
        loc=0;
        input_ready=!TRUE;
    }
}
```

```
        if(!paused && auto_stop && check_time<s_time()){
            SCOPE_init(SCOPE_ADDR);
            OSC_set_power(cur_power);
            value=SCOPE_f_height(edge,range/20,15);
            _settextcolor(RED);
            _settextposition(BLOCK4+2,1);
            sprintf(text,"Signal=%f          ",value);
            _outtext(text);
            if(value>2*noise){
                cur_freq-=freq_step;
                print[C_FREQUENCY]=TRUE;
                beep();
                if(find_q){
                    num_steps=0;
                    measure_q=TRUE;
                }
                else{
                    paused=TRUE;
                    _settextcolor(RED+BLINK);
                    _settextposition(1,1);
                    _outtext("PAUSED");
                    /*OSC_close();*/
                    /*SCOPE_close();*/
                }
                if(*stop_file !=0){
                    paused=!TRUE;
                    if((sfile=fopen(stop_file,"a"))!=NULL){
                        fprintf(sfile,"%.10g %g\n",cur_freq,value);
                        fclose(sfile);
                    }
                    else{
                        _settextcolor(RED+BLINK);
                        _settextposition(BLOCK4+3,1);
                        sprintf(text,"ERROR -- FILE NOT OPENED");
                    }
                    cur_freq+=freq_step;
                    SCOPE_set_range(edge-range/10,range);
                }
            }
            check_time=s_time()+check_int;
            /*SCOPE_close();*/
        }
        if(cur_freq<20 && cur_freq>.01 && !paused){
            print[C_FREQUENCY]=TRUE;
            cur_freq+=freq_step;
        }
    }
    if(do_noise)
        cur_mode=old_mode;
    par_file = fopen ("c:\\lw\\programs\\modes.par", "w+b");
    fwrite ((char *)mode,sizeof mode,1,par_file);
        fwrite ((char *)&noise,sizeof(double),1,par_file);
        fwrite ((char *)&check_int,sizeof(long),1,par_file);
        fwrite ((char *)&freq_step,sizeof(double),1,par_file);
```

```
            fwrite ((char *)&range,sizeof(double),1,par_file);
            fwrite ((char *)&edge,sizeof(double),1,par_file);
            fwrite ((char *)&cur_mode,sizeof(int),1,par_file);
            fwrite (f_name,30,1,par_file);
            fwrite (trans_name,80,1,par_file);
            fwrite (direct,80,1,par_file);
        fclose (par_file);
        _setvideomode(_DEFAULTMODE);

}
/* performs scans across all modes */
/*This function systematically changes the frequency from start value to end vale*/
/*and measures the voltage level on the oscilliscope*/
#define DELAY2 2
#define MAX_LINE 63
Scan(mode,direct,f_name,edge,range,start,q_val,level,noise)
      double noise, level;  int start;
      struct mod {
            double freq;
            float power;
            float q;        /* actally scan width */
            float delay;
            int scan_res;
            int envelope;
            } *mode;
      double edge,range; char *f_name, *direct;  float *q_val;
{
      struct sum {
            float sum;
            float sumsqr;
            float min;
            float max;
            int num;
      } t[3];
      int no_temp, time_t ltime;    FILE *output, *sum_dat;
      int l;  char text[150]; double coax; int n,m,m2,key, pt, abort;
      double start_freq, cur_freq,freq_span,height, temp0;   char t_name[100];
      double x[MAX_POINTS], y[MAX_POINTS], f0,y3db,xhigh,xlow;
      FILE *printer;     static int print_line=0;    int printing;
      abort=1;
      key=0;
      no_temp=!TRUE;
      if((printer=fopen("prn","w"))!=NULL){
            if(print_line>MAX_LINE-7){
                  time(&ltime);
                  fprintf(printer,"%c\n\t\t%s\n\n\n\n",12,ctime(&ltime)); /* form feed*/
                  print_line=6;
            }
            else if(print_line==0){
                  time(&ltime);
                   fprintf(printer,"\n\t\t%s\n\n\n\n",ctime(&ltime)); /* form feed*/
                  print_line=6;
            }
            printing=TRUE;
```

```
        fprintf(printer,"\n\t%8.8s.*\n",f_name);
        sprintf(text,"Noise=%.6g V",noise);
        fprintf(printer,"\t   %-34s0-time=%.6g us\n\n",text,edge*1e6);
        print_line+=4;
        fclose(printer);
}
else{
        _settextcolor(YELLOW+BLINK);
        _outtext("PRINTER NOT CONNECTED--DATA NOT PRINTED");
        pause(5);
        printing=!TRUE;
}
OSC_init(OSC_ADDR);
OSC_set_power((double)mode[start%MAX_MODE].power);
SCOPE_init(SCOPE_ADDR);
OSC_set_power((double)mode[start%MAX_MODE].power);
TEMP_init(TEMP_ADDR);
OSC_set_power((double)mode[start%MAX_MODE].power);
for(n=start;abort>-1 && n<MAX_MODE+start;n++){
        abort=1;
        if(mode[n%MAX_MODE].freq>.01 && mode[n%MAX_MODE].freq
                                <20.0 && mode[n%MAX_MODE].scan_res>0){
                sprintf(t_name,"%s%s.M%i",direct,f_name,n%MAX_MODE);
                if(SCOPE_set_range(edge+mode[n%MAX_MODE].delay,
                                (edge+2*mode[n%MAX_MODE].delay)/900)<0)
                        _outtext("error!!");
                _settextposition(BLOCK4,1);
                if((output=fopen(t_name,"a"))==NULL){
                        _settextcolor(YELLOW+BLINK);
                        _outtext("INVALID FILE NAME");
                        pause(5);
                        abort=-2;
                }
                else{
                        sprintf(text,"output file:%s\n",t_name);
                        _settextcolor(YELLOW);
                        _outtext(text);
                }
                _outtext("Enter coax height (<0 to abort): ");
                scanf("%lf",&coax);
                if(coax<0)
                        abort=-2.0;
                else{
                        start_freq=mode[n%MAX_MODE].freq-mode[n%MAX_MODE].q/2;
                        OSC_set_freq(start_freq);
                        OSC_set_power((double)mode[n%MAX_MODE].power);
                }
                for(m=0;!no_temp && m<3;m++){
                                t[m].sum=t[m].sumsqr=0.0;
                                t[m].num=0;
                                t[m].max=-(t[m].min=1e5);
                }
                if(!no_temp && TEMP_read(T_CONTROL)<0){
                                _clearscreen(_GCLEARSCREEN);
```

```
            printf("Cannot read temp.  Proceed anyway (y/n)?");
            m=toupper((int)getch());
            if(m=='N')
                abort=-2;
            else{
                no_temp=TRUE;
                printf("\nEnter temperature: ");
                scanf("%lf",&temp0);
                t[1].sum=t[1].sumsqr=
                        t[1].max=t[1].min=temp0;
                t[1].num=1;
            }
}
if(abort>0){
    sprintf(t_name,"%s%.5s.M%i",direct,f_name,n%MAX_MODE);
    sum_dat=fopen(t_name,"a");
    fprintf(output,"/* power=%.3g\n",mode[n%MAX_MODE].power);
    fprintf(output,"/* noise=%e\n",noise);
    sprintf(text,"/* power=%.3g",mode[n%MAX_MODE].power);
    _settextposition(BLOCK4+1,1);
    _outtext(text);
    for(l=0;l<10;l++){
        while(_bios_keybrd(_KEYBRD_READY)!=0){
            if((key=_bios_keybrd(_KEYBRD_READ))==0x11b)
                abort=-1;
            else if(key==0x1C0D)
                abort=0;
        }
        pause(1);
    }
    start_freq=mode[n%MAX_MODE].freq-mode[n%MAX_MODE].q/2;
    SetAxAuto (0, 0);
    SetAxRange (0, start_freq, start_freq+mode[n%MAX_MODE].q, 10);
    SetAxAuto (1, 2);
    SetAxRange (1, 0.0, 1.0, 10);
    SetFrmColor (13);
    SetTtlColor (11);
    SetGrdColor (9);
    SetLblColor (9);
    SetCurv2D (2);
    sprintf(t_name,"%s%s.M%i",direct,f_name,n%MAX_MODE);
    SetTitle (t_name);
    sprintf(text,"f0=%.9f",mode[n%MAX_MODE].freq);
    SetAxName (0, text);
    for(pt=0;pt<=mode[n%MAX_MODE].scan_res && abort>0;pt++){
        cur_freq=start_freq+mode[n%MAX_MODE].q*
                ((double)pt/mode[n%MAX_MODE].scan_res);
        OSC_set_freq(cur_freq);
        for(m=0;!no_temp && m<3;m++){
            temp0=TEMP_read(m-1);
            if(temp0>-1.0){
                if(t[m].min>temp0)
                    t[m].min=temp0;
                if(t[m].max<temp0)
```

```
                    t[m].max=temp0;
                t[m].sum+=temp0;
                t[m].sumsqr+=temp0*temp0;
                t[m].num++;
            }
        }
        pause(DELAY2);
        cur_freq=OSC_read_freq();
        height=SCOPE_height(mode[n%MAX_MODE].envelope)-noise;
        fprintf(output,"%.8e %.8e\n",cur_freq,height);
        x[pt]=cur_freq;
        y[pt]=height;
        SetPlotMode (0);
        SetCrvColor ((n%MAX_MODE+1)%15+1);
        SetPointStyle (5);
        GrfCurv2D(&cur_freq,&height,1);
        while(_bios_keybrd(_KEYBRD_READY)!=0){
            if((key=_bios_keybrd(_KEYBRD_READ))==0x11b)
                abort=-1;
            else if(key==0x1C0D)
                abort=0;
        }
        if(abort<=0)
            printf("ABORTING SCAN\n");
    }
}
if(abort>-2){
    for(m=0;!no_temp && m<3;m++){
            temp0=TEMP_read(m-1);
            if(temp0>-1.0){
                if(t[m].min>temp0)
                    t[m].min=temp0;
                if(t[m].max<temp0)
                    t[m].max=temp0;
                t[m].sum+=temp0;
                t[m].sumsqr+=temp0*temp0;
                t[m].num++;
            }
    }
    for(m=0;m<3;m++){
        if(t[m].num>0){
            t[m].sum/=t[m].num;
            t[m].sumsqr=sqrt(t[m].sumsqr/t[m].num-t[m].sum*t[m].sum);
            switch(m){
                case 0: sprintf(text,"/* Heater=");break;
                case 1: sprintf(text,"/* Temp =");break;
                case 2: sprintf(text,"/* Temp2 =");break;
            }
            if(no_temp)
                fprintf(output,"%s %g\n",text,t[m].sum);
            else
                fprintf(output,"%s%g +-%g (%g -> %g)%ireadings\n",text,t[m].sum
                        ,t[m].sumsqr,t[m].min,t[m].max,t[m].num);
        }
```

```
}
q_val[n%MAX_MODE]=-1;
if(pt>mode[n%MAX_MODE].scan_res/3 && pt>3){
      smooth(x,y,pt-1,level);
      meas_q(x,y,pt,q_val+n%MAX_MODE,&f0,&xlow,&xhigh,&y3db,0);
      if(f0>0)
            mode[n%MAX_MODE].freq=f0;
      fprintf(output,"/* q= %E\n/* f0= %.12E\n",q_val[n%MAX_MODE],f0);
      fprintf(output,"//nc\n%e %e\n%e %e\n",xlow,y3db,xhigh,y3db);
      fprintf(sum_dat,"/* %s\n%.4g %.11e %.11e %.11e\n",f_name,
            t[1].sum,q_val[n%MAX_MODE]
            ,6.283185307*f0*1e9/q_val[n%MAX_MODE],f0);
}
fclose(output);
fclose(sum_dat);
SetDisplayMode (0);
DeletePort (0);
/*******send data to printer *****/
if(abort>-2 && (printer=fopen("prn","w"))!=NULL){
  if(print_line>MAX_LINE-4){
     time(&ltime);
     fprintf(printer,"%c\n\t\t%s\n\n\n\n",12,ctime(&ltime)); /* form feed*/
     print_line=6;
  }
  for(m2=0;m2<3;m2++){
     m=2-m2;
     switch(m2){
        case 0:
           sprintf(text,"M%-19iF0=%.9fGHz",n%MAX_MODE,f0);
           fprintf(printer,"\t%-54s",text);
           break;
        case 1:
           sprintf(text,"Width=%.5fMHz",mode[n%MAX_MODE].q*1000);
           fprintf(printer,"\t  %-17s",text);
           sprintf(text,"Level=%.3fdBm",mode[n%MAX_MODE].power);
           fprintf(printer,"%-17s",text);
           sprintf(text,"delay=%.4fus",mode[n%MAX_MODE].delay*1e6);
           fprintf(printer,"%-17s",text);
           break;
        case 2:
           fprintf(printer,"\t  Resolution=%-6iEnvelope=%-25i",mode
              [n%MAX_MODE].scan_res,mode[n%MAX_MODE].envelope);
           break;
     }
     switch(m){
              case 0: fprintf(printer,"Pow=");break;
              case 1: fprintf(printer,"C=");break;
              case 2: fprintf(printer,"T=");break;
     }
     if(no_temp)
              fprintf(printer," %g",t[m].sum);
     else
              fprintf(printer," %.4f+-%.4f",t[m].sum
                       ,t[m].sumsqr);
```

```
                        switch(m){
                                case 0: fprintf(printer,"\%\n");break;
                                case 1: fprintf(printer,"nF\n");break;
                                case 2: fprintf(printer,"K\n");break;
                        }
                        print_line++;
                    }
                    fprintf(printer,"\t  Q=%-15.6gD=%-15.6gAmp=%-13.6gCoax=%-
                            12.3g\n\n",q_val[n%MAX_MODE]
                           ,6.283185307*f0*1e9/q_val[n%MAX_MODE],y3db*1.414213562,coax);
                    print_line+=2;
                    fclose(printer);
                   /*send data to floppy*/
                   m=0;
                   while(floppy(direct,f_name,n)<0 && m!='x' && m!='X'){
                      _outtext("ERROR: file not saved to floppy\n  Press any key to try again. (x to
                               abort)");
                      m=getch();
                   }
                }
            }
        }
    SCOPE_set_range(edge-range/10,range);
return 0;
}
/*smooth curve to be able to more accurately measure the peak width*/
int smooth(x,y,pts,level)
int pts; /* pts=# of points */
double x[100],y[100];
double level;
{
  int n1,n2,n;
  double m;
    for(n1=0;n1<pts-2;n1++)
        if( (n1==0 || y[n1]<y[n1-1]) && y[n1]<y[n1+1]){
            for(n2=n1+2;n2<pts && n2<n1+pts*level;n2++)
                if( (n2==pts-1 || y[n2]<y[n2-1]) && y[n2]<y[n2+1]){
                    m=(y[n1]-y[n2])/(x[n1]-x[n2]);
                    for(n=n1+1;n<n2;n++)
                        y[n]=y[n1]-m*(x[n1]-x[n]);
                    n1=n2;
                }
            n1=n2-1;
        }
    return 0;
}
/* measure Q of the mode/*
/*actally, scan data for peak (vs. frequency), measure width of the peak*/
/*This routine must be careful not to misinterpret noise as a peak*/
meas_q(x,y,pt,q_val,f0,xlow,xhigh,y3db,peak_number)
int peak_number; /* defines which peak to measure */
int pt;
double *x,*y;
```

```
double *f0;
double *xlow,*xhigh,*y3db;
float *q_val;
{
    int l;
    double d1,d2;
    int peak_found,ismin;
    double ymax,yhigh,ylow,alpha,deltax;
    int n,n1,n2,iend,imax;
            *f0=-1;
            /*********************************************************/
            /*find q and reset frequency to new ceter frequency     */
            /*********************************************************/
                ymax=y[0];
                imax=0;
                /*****find local maximum nearest center*****/
                peak_found=!TRUE;
                for(n1=0;n1<=(pt-1)/2-1 && !peak_found;n1++){
                    for(n2=-1;n2<=1 && !peak_found;n2+=2){
                        /* check point y[(pt-1)/2+n1*n2] for being a maximum */
                        imax=(pt-1)/2+n1*n2;
                        ymax=y[imax];
                        peak_found=TRUE;
                        for(l=-(pt-1)/20-1;l<=(pt-1)/20+1 &&peak_found;l++)
                            if(imax+l>=0 && imax+l<pt && y[imax+l]>y[imax])
                                peak_found=!TRUE;
                    }
                }
                /*find minimum peak_number left of center ******/
                n2=(peak_number<0)?-1:1;
                n=abs(peak_number);
                for(n1=imax+n2;n>0 && n1>=0 && n1<pt;n1+=n2){
                        /* check point y[n1] for being a maximum */
                        imax=n1;
                        ymax=y[imax];
                        peak_found=TRUE;
                    for(l=-(pt-1)/20-1;l<=(pt-1)/20+1 &&peak_found;l++)
                        if(y[imax+l]>y[imax])
                            peak_found=!TRUE;
                    if(peak_found && n<abs(peak_number)){
                        peak_found=!TRUE;
                        n--;
                    }
                }
                *y3db = .707106781 * ymax;      /* -3 dB from peak*/
                if (!peak_found){
                    *q_val=-1;
                    *f0=(x[0]+x[pt-1])/2;
                    *xhigh=x[0];
                    *xlow=x[pt-1];
                }
                else{    /*find true center and true maximum */
                    *f0=0;
                    if(y[imax]==y[imax-1] && y[imax]==y[imax+1])
```

```
            *f0=(2*x[imax]+x[imax-1]+x[imax+1])/4;
        else if(y[imax]==y[imax-1])
            *f0=(x[imax]+x[imax-1])/2;
        else if(y[imax]==y[imax+1])
            *f0=(x[imax]+x[imax+1])/2;
        else{
            *f0=(x[imax]+x[imax-1]) / (2*(1+(y[imax]-y[imax-1])
                /(y[imax]-y[imax+1])))
            +(x[imax]+x[imax+1])/(2*(1+(y[imax]-y[imax+1])/(y[imax]-
                y[imax-1])));
        }
    /* IS THIS REASONABLE????  Just use max point */
/*      d2=(x[imax]-*f0);
        d2=d2*d2;
        if(*f0<x[imax]){
            d1=(x[imax-1]-*f0);
            d1=d1*d1;
            ymax=y[imax]*y[imax-1]*(d2-d1)/(d2*y[imax]-d1*y[imax-1]);
        }
        else if(*f0>x[imax]){
            d1=(x[imax+1]-*f0);
            d1=d1*d1;
            ymax=y[imax]*y[imax+1]*(d2-d1)/(d2*y[imax]-d1*y[imax+1]);
        }
        else
            ymax=y[imax];
        *y3db = .707106781 * ymax;   */     /* -3 dB from peak*/
    /*    if(ymax>1.1*y[imax])  */
    /*        ymax=y[imax];  if this estimate is too far from point, ignore it */
        /*find left 3db or min*/
        ismin=!TRUE;
        for(n1=imax;!ismin && y[n1]>*y3db && n1>=0;n1--){
            ismin=TRUE;
            for(n2=n1-1;n2>=0 && n2>=n1-(pt-1)/20-1;n2--)
                if(y[n2]<y[n1])
                    ismin=!TRUE;
        }
        if(ismin){   /*no 3db point*/
            n1=((SHIFT_MIN-1)*(n1+1)+imax)/SHIFT_MIN+(pt-1.0)/20.0+1;
                        /*move slightly away from true minimum */
            if(n1>=imax){
                *xlow=*f0;
                ylow=ymax;
            }
            else{
                *xlow=x[n1];
                ylow=y[n1];
            }
        }
        else{
            ylow=*y3db;
            *xlow=x[n1]+(x[n1+1]-x[n1])*(*y3db-y[n1])/(y[n1+1]-y[n1]);
        }
        /*find right 3db or min*/
```

```
                ismin=!TRUE;
                for(n1=imax;!ismin && y[n1]>*y3db && n1<=(pt-1);n1++){
                        ismin=TRUE;
                        for(n2=n1+1;n2<=(pt-1) && n2<=n1+(pt-1)/20+1;n2++)
                                if(y[n2]<y[n1])
                                        ismin=!TRUE;
                }
                if(ismin){   /*no 3db point*/
                        n1=((SHIFT_MIN-1)*(n1-1)+imax)/SHIFT_MIN-(pt-1.0)/20.0-1;
                                        /*move slightly away from true minimum */
                        if(n1<=imax){
                                *xhigh=ymax;
                                yhigh=*f0;
                        }
                        else{
                                *xhigh=x[n1];
                                yhigh=y[n1];
                        }
                }
                else{
                        n1--;
                        yhigh=*y3db;
                        *xhigh=x[n1]+(x[n1+1]-x[n1])*(*y3db-y[n1])/(y[n1+1]-y[n1]);
                }
                alpha=ylow/ymax;
                if(alpha>.99)
                        *xlow=0;
                else
                        *xlow = *f0 - (*f0-*xlow) * sqrt(.414213562 * alpha / (1 - alpha));
                alpha=yhigh/ymax;
                if(alpha>.99)
                        *xhigh=0;
                else
                        *xhigh = *f0 + (*xhigh-*f0) * sqrt(.414213562 * alpha / (1 - alpha));
                if(*xlow==0 && *xhigh==0){
                        *xhigh=*f0;
                        *xlow=*f0;
                }
                if(*xlow==0)
                        *xlow=*f0+*f0-*xhigh;
                if(*xhigh==0)
                        *xhigh=*f0+*f0-*xlow;
                deltax=(*xhigh-*xlow);
                *f0=(*xhigh+*xlow)/2;
                if(deltax==0)
                        *q_val=-1;
                else
                        *q_val=*f0/deltax;
        }
}

/*****************************************************/
/*Routines used to communicate to the Oscilliscope                */
/*****************************************************/
```

```
extern char cursor[4];
extern double data[3][600]; /* buffer for use by all subroutines */
#define N_AVG 10

/*****************************************************/
/*Read the scope and measure the height of channel 2*/
/*using an envelope of (count) points          */
/*and digitize curve into 128 points           */
/*****************************************************/
double SCOPE_height(int count)
{
      long time0;
      int points,n;
      double height,b,x,x_sqr,y,y_sqr,xy,t;
      if(count<=0)
            count=32;
      SCOPE_digitize(2,count,128);
      time0=s_time();
      SCOPE_ready(time0,300);
      points=SCOPE_read();
      x=x_sqr=y=y_sqr=xy=0;
      for(n=0;n<points;n++){
            x+=n;
            x_sqr+=n*n;
            y+=(t=data[2][n]-data[1][n]);
            y_sqr+=t*t;
            xy+=n*t;
      }
      y/=2;
      y_sqr/=4;
      xy/=2;
      b=0;
      if(points*x_sqr-x*x!=0)
            b=(points*xy-x*y)/(points*x_sqr-x*x);
      height=(y-b*x)/points;
      return height;
}
/*set zero time to falling edge of the pulse*/
double SCOPE_find_edge(int points,int col)
{
      int n,m;
      double diff,max_diff;
      int edge;

      for(n=0;n<N_AVG;n++)
            diff=data[col][n]-data[col][n+N_AVG];
      max_diff=diff;
      edge=N_AVG;
      for(n=N_AVG;n<points-N_AVG;n++){
            /*printf("data[%i]=%g\n",n,data[col][n]);*/
            diff+=2*data[col][n]-data[col][n-N_AVG]-data[col][n+N_AVG];
            if(diff>max_diff){
                  max_diff=diff;
                  edge=n+1;
```

```
            }
        }
        return data[0][edge];
}

int SCOPE_init(int addr)
{
        scope_addr=addr;
        GEN_init(addr);
        GEN_write(scope_addr,"*CLS",4);
        return ibsta;
}

int SCOPE_close()
{
        GEN_loc(scope_addr);
        return ibsta;
}

/*set range of time axis on scope*/
int SCOPE_set_range(double start, double range)
{
        char cmd[30];

        GEN_write(scope_addr,":TIM:REF LEFT",13);
        sprintf(cmd,":TIM:DEL %E",start);
        GEN_write(scope_addr,cmd,strlen(cmd));
        sprintf(cmd,":TIM:RANG %E",range);
        GEN_write(scope_addr,cmd,strlen(cmd));
        return ibsta;
}

/*wait until scope is ready to respond.  Usually, this is done while waiting for
/*the scope to finish a measurement*/
SCOPE_ready(long time0,int timeout)
{
  long timeout2;
  int n,m;
     timeout2=(long)timeout*1000;
     if(SCOPE_rsp()==1)
          n=0;
     else
          n=-1;
     while (n==-1 && s_time() - time0 < timeout2)  {
          pause(1);
          if(SCOPE_rsp()==1)
               n=0;
          else
               if(_bios_keybrd(_KEYBRD_READY)!=0)
                    return(_bios_keybrd(_KEYBRD_READ));
     }
     if(n==-1)
          beep();
     return n;
```

```
}
/*get serial poll-request information from scope*/
int SCOPE_rsp()
{
  char com[30];
  int m;
      GEN_timeout(scope_addr,15); /* set timeout=100sec */
      while(GEN_write(scope_addr,"*OPC?",5)<0);
      GEN_read(scope_addr,com,20);
      sscanf(com,"%i",&m);
      if(ibsta<0)
           m=-1;
      return m;
      GEN_timeout(scope_addr,13); /* set timeout=10sec */
}

/*****************************************************/
/*digitize oscilliscope channel # (channel)       */
/*    acquire after looking at (count) measurements*/
/*    digitize resulting curve into (points) points*/
/*****************************************************/
int SCOPE_digitize(int channel,int count,int points)
{
char com[30];
      sprintf(com,":ACQUIRE:COUNT %i",count);
     GEN_write(scope_addr,":ACQUIRE:TYPE ENV",17);
     GEN_write(scope_addr,com,strlen(com));
     sprintf(com,":ACQUIRE:POINTS %i",points);
     GEN_write(scope_addr,com,strlen(com));
     GEN_write(scope_addr,":SYSTEM:HEADER OFF",18);
     sprintf(com,":WAVEFORM:SOURCE WMEMORY%i",channel);
     GEN_write(scope_addr,com,strlen(com));
     GEN_write(scope_addr,":WAVEFORM:FORMAT WORD",21);
     sprintf(com,":DIGITIZE CHANNEL%i",channel);
     GEN_write(scope_addr,com,strlen(com));
     return 0;
}

int SCOPE_read()
{
      unsigned char cmd[300];
     char *names;
     int n,m,points;
     float x_inc,x_org,x_ref,y_inc,y_org,y_ref,y_range;
     int num,num1,num2;
          GEN_write(scope_addr,":WAVEFORM:PRE?",14);
          GEN_read(scope_addr,cmd,298);
          GEN_write(scope_addr,":WAVEFORM:DATA?",15);
          sscanf(cmd,"%*i,%*i,%i,%*i,%f,%f,%f,%f,%f,%f,%f",&points,&x_inc,&x_org,
                &x_ref,&y_inc,&y_org,&y_ref,&y_range);
          GEN_read(scope_addr,cmd,2);
          num=cmd[1]-'0';
          GEN_read(scope_addr,cmd,num);
          cmd[num]=0;
```

```
            sscanf(cmd,"%i",&points);
            points/=4;
            if(points>512){
                  beep();
                  printf("ERROR: points=%i\n",points);
                  points=512;
            }
            for(n=0;n<points;n++){
                  GEN_read(scope_addr,cmd,2);
                  data[0][n]=(n-x_ref)*x_inc+x_org;
                  data[1][n]=(cmd[0]*256.0+cmd[1]-y_ref)*y_inc+y_org;
            }
            for(n=0;n<points;n++){
                  GEN_read(scope_addr,cmd,2);
                  data[2][n]=(cmd[0]*256.0+cmd[1]-y_ref)*y_inc+y_org;
            }
      return points;
}
/*initialize scope to standard state*/
SCOPE_setup()
{
      GEN_write(scope_addr,":BLANK FUNC2",12);
      GEN_write(scope_addr,":BLANK CHAN1",12);
      GEN_write(scope_addr,":DISP:PERS .3",13);
      GEN_write(scope_addr,":FUNC1:MAX CHAN2",16);
      GEN_write(scope_addr,":DISP:VMAR ON",13);
      GEN_write(scope_addr,":DISP:TMAR ON",13);
      GEN_write(scope_addr,":VIEW CHAN3",11);
      GEN_write(scope_addr,":VIEW FUNC1",11);
      GEN_write(scope_addr,":VIEW CHAN2",11);
      GEN_write(scope_addr,":WAV:FORM WORD",14);
      GEN_write(scope_addr,":MEAS:PREC FINE",15);
      return 0;
}

int SCOPE_measure(char *channel)
{
      char com[30];
      sprintf(com,":MEAS:SOUR %s",channel);
      GEN_write(scope_addr,com,strlen(com));
      return ibsta;
}
double SCOPE_c_height(double time)
{
      double height;
      char com[40];
      SCOPE_measure("CHAN2"); /* do measurements on channel 2 (signal) */
            sprintf(com,":MEAS:VTIM? %G",time);
            GEN_write(scope_addr,com,strlen(com));
            GEN_read(scope_addr,com,38);
            sscanf(com,"%lf",&height);
      return height;
}
/*this function is used to clear the screen on the scope*/
```

```
int SCOPE_reset(function)
{
    GEN_write(scope_addr,":FUNC1:MAX CHAN2",17);
    return ibsta;
}
/*measure height of one of the function channels (ususally maximum or minimum functions)*/
double SCOPE_f_height(time,range,points)
double time;
double range;
int points;
{
    double height;
    char com[40];
    double x,x_sqr,y,y_sqr,xy;
    int n;
    double b;
    if(points<=0)
        points=10;
    SCOPE_measure("FUNC1"); /* do measurements on max of channel 2 (signal) */
    x=x_sqr=y=y_sqr=xy=0;
    for(n=0;n<points;n++){
        sprintf(com,":MEAS:VTIM? %G",time+n*range/points);
        GEN_write(scope_addr,com,strlen(com));
        GEN_read(scope_addr,com,38);
        sscanf(com,"%lf",&height);
        /*printf("f_ht:%g at %g '%s'\n",height,time+n*range/points,com);*/
        x+=n;
        x_sqr+=n+n;
        y+=height;
        y_sqr+=height*height;
        xy+=n*height;
    }
    b=0;
    if(points*x_sqr-x*x==0)
        printf("%i x=%f,x2=%f,y=%f,y2=%f,xy=%f",n,x,x_sqr,y,y_sqr,xy);
    else
        b=(points*xy-x*y)/(points*x_sqr-x*x);
    height=(y-b*x)/points;
    return height;
}
```

[1] N.-C. Yeh, U. Kriplani, W. Jiang, D. S. Reed, D. M. Strayer, J. B. Barner, B. D. Hunt, M. C. Foote, R. P. Vasquez, and A. Kussmaul, Phys. Rev. B **48**, 9861 (1993); U. Kriplani, N.-C. Yeh, W. Jiang, D. S. Reed, A. Gupta, and A. Kussmaul, *Layered Superconductors: Fabrication, Properties, and Applications*, edited by D. T. Shaw *et al.*, MRS Symposia Proceedings No. 275 (Materials Research Society, Pittsburgh, 1992), p.795.