LBL-31143

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA

Materials & Chemical
Sciences Division

Studies of Nonlinear Electrodynamics of High-Temperature Superconductors

Q.-C.H. Lam
(Ph.D. Thesis)

August 1991

Prepared for the U.S. Department of Energy under Contract Number DE-AC03-76SF00098

DISCLAIMER

LBL--31143

DE92 000814

STUDIES OF NONLINEAR ELECTRODYNAMICS OF HIGH-TEMPERATURE SUPERCONDUCTORS*

Quan-Chiu Harry Lam

Ph.D. Thesis

Department of Physics
University of California, Berkeley, CA 94720

and

Materials Sciences Division
Lawrence Berkeley Laboratory, Berkeley, CA 94720

August 1991

---

*This work was supported in part by the Director, Office of Energy Research, Office of Basic Energy Sciences, Materials Sciences Division of the U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

Studies of Nonlinear Electrodynamics of High-Temperature Superconductors

by

Quan-chiu Harry Lam

Studies of Nonlinear Electrodynamics of High-Temperature Superconductors.

by

Quan-chiu Harry Lam

ABSTRACT

Nonlinear electrodynamics of high-T_c superconductors are studied both theoretically and experimentally. For powdered samples, a novel model is presented in which the metallographically observed superconducting grains in the powder are modeled as superconducting current loops of various areas with weak links. Surprising harmonic generation behavior in an ac field, $H_1 \cos(\omega t)$, is predicted by the model; the power at high harmonics show sharp dips almost periodic in a superposing dc magnetic field, revealing flux quantization in the prototype loops in the model. Such oscillation of the harmonic power in dc magnetic field, $P_{nf}(H_{dc})$, is indeed experimentally observed in powdered $YBa_2Cu_3O_7$. Other experimental aspects also agree with model predictions. For bulk sintered cylindrical samples, a generalized critical state model is presented. In this model, the nonlinear electrodynamics are due to flux-pinning, somewhat similar to low-temperature type-II superconductors, but with a more generalized critical current densities' dependence on magnetic field — $J_c(H) \sim H_{local}^{-\beta}$, with β being an adjustable parameter. Experiments in ac and dc magnetic fields on a sintered cylindrical rod of $YBa_2Cu_3O_7$ yield *unambiguous* evidence of independent inter- and intragranular contributions to the complex harmonic permeability $\tilde{\mu}_n = \mu'_n - i\mu''_n$. For example, two peaks in $\mu''_1(H_1)$ are clearly observed, which signify ac absorption by the inter- and intragranular supercurrents, respectively. These data, together with $P_{nf}(H_{dc})$, are explained very well quantitatively by the generalized critical state model, yielding a dependence on magnetic field of $J_c(H) \sim H_{local}^{-2}$ for the intergranular component, a steeper field dependence than for conventional type-II superconductors. Temperature-

dependence measurements reveal that, while the intragranular supercurrents disappear at $T_c \geq 91.2$ K, the intergranular supercurrents disappear at $T \geq 86.6$ K. This result is, to our knowledge, the first clear measurement of the phase-locking temperature of the 3-D matrix formed by $YBa_2Cu_3O_7$ grains, which are in electrical contact with one another through weak links.

Similar experimental data on a $YBa_2Cu_3O_7$ thin-film and a $Bi_2Sr_2CaCu_2O_8$ single crystal are also presented. In both samples, only one supercurrent component is observed. The data are explained by the generalized critical state model and estimates of critical current densities are obtained.

Professor Carson D. Jeffries
Chairman, Dissertation Committee

To my parents, Wan-fa and Fung-kwong Lam.

ACKNOWLEDGEMENTS

I would like to thank Professor Carson D. Jeffries for his continued advice, support, and encouragement throughout the course of this work. I am particularly grateful to him for his profound scientific insights and his patience in teaching me how to do research. Working with him has truly been a fruitful and inspiring experience.

I am very grateful to Professor Alan Portis, who has spent many hours of discussions with me, giving me many useful comments and suggestions. In addition, I also thank Professor Alex Zettl and Professor Eugene Haller for their precious time to serve as members of my qualifying exam as well as dissertation committees. The same amount of thanks should also go to Professor Leo Falicov for useful discussions on magnetism in superconductors and his service in my qualifying exam committee. I acknowledge Professor John Clarke for important suggestions concerning the modeling of Josephson junctions, and Dr. James Crutchfield for discussions on nonlinear dynamics and computer modeling. Thanks also go to the members of the Zettl group, in particular, Lincoln Bourne, Angelica Behrooz, Michael Crommie, and Gabriel Briceno, who have provided me with many high quality samples and let me use some of their experimental apparatus. I also thank Dr. Paul Berdahl, Dr. Richard Russo, and Mr. Ron Reade for providing me with the high-quality Y-Ba-Cu-O thin-film sample.

Many fellow graduate students have also given me much assistance at various stages. My former colleague Youngtae Kim first suggested the idea of generalizing the critical state model. My classmates, Wei Chen, Andrew Cleland, Non Fan, Jim Hanson, Larry Wald, Ning Wang and Xu-dong Xiao, have given me many useful comments and suggestions regarding my oral presentations in conferences, qualifying exam, and seminars.

The excellent support facilities such as the machine shop and the electronic shop also deserve great thanks. The efficiency, dedication, and skills of their staff have definitely played a crucial role to the success of our whole Physics Department. I

particularly acknowledge the help and friendship of Pete Miller, whose efficiency in repairing electronics has enabled me to finish many of my projects on schedule.

I thank the support given by both the Office of Naval Research and the Department of Energy to this work.

Finally I would like to express my deepest gratitude to my parents, my brother Louis, and sisters Jessie, Olivia, Eva and Shirley. Their love of me has always been my greatest asset. It is not an exaggeration to say that I owe this entire work to their selfless moral support.

Table of Contents

List of Tables

For number sequence only.

Chapter 1 Introduction

The discovery of high-temperature copper-oxide superconductors by Bednorz and Müller [1] triggered an enormous amount of international research effort to understand the materials' nature and properties, and to put the materials into application. However, for application purposes the efforts have encountered many difficulties, including unusually small critical current densities in ceramic samples [2], and high sensitivity of the critical current to small magnetic fields. These unusual properties have been ascribed by many to the granular nature of the materials, and they bear some resemblance to those of the low-temperature granular superconductors. The latter granular materials, though, have usually been intentionally fabricated to exhibit granularity for the sake of experimental studies. For these materials, granularity is usually only of academic interests. But for high-temperature superconductors, granularity *may* well be something we will have to deal with, and live with, in practical applications.

To help understand these unusual properties of high-temperature superconductors, we study the nonlinear electrodynamics of the materials. The nonlinearity may be due to the granularity of the materials, such as Josephson weak links in the samples [3][4][5][6], or, from the point of view of conventional type-II superconductors, magnetic hysteresis [7][8][9]. To be more specific, we experimentally study the materials through the harmonics they generate when they are driven by a low-frequency ac magnetic field, while being in a superconducting state. High harmonics are particularly interesting because they are extremely sensitive to the details of the materials' electrodynamical properties such as the dependence of the critical currents on magnetic field. Thus they provide very severe tests to the theoretical models which we will use to help understand the experimental data. But before we go into details of the theoretical models and the experiments, let us review some background information in this chapter.

1.1 High-Temperature Superconductors

Since much has been published in the literature about the high-temperature copper-oxide type of superconductors [10][11][12], we will only briefly go over some of the general properties of the two particular compounds relevant to us in this thesis, namely, $YBa_2Cu_3O_{7-\delta}$ and $Bi_2Sr_2CaCu_2O_8$.

Y-Ba-Cu-O. Y-Ba-Cu-O compounds were first reported by M.K. Wu *et al* [13] to be superconducting at an amazingly high temperature: 92 K. The relevant compound responsible was later identified as $YBa_2Cu_3O_{7-\delta}$. The crystallographic structure of this compound is of the perovskite type. The structure of a prototypical perovskite compound $BaTiO_3$ is shown in Figure 1.1.1.

The compound $YBa_2Cu_3O_{7-\delta}$ exists in tetragonal and orthorhombic phases [12]. The tetragonal $YBa_2Cu_3O_{7-\delta}$ (Figure 1.1.2(a)) is stable above about 650°C with $\delta >$ 0.5. The unit cell dimensions are $a = b = 3.90$ Å and $c = 11.94$ Å; the structure may be visualized as being derived from three prototype perovskite unit cells stacked one above the other along the c-axis. This tetragonal phase can also be obtained at room temperature by quenching from above $\sim$ 700°C, and it is found to be semiconducting. However, if instead the temperature is slowly lowered from above $\sim$ 700°C, the compound will undergo a second order phase transition from the tetragonal to an orthorhombic phase (Figure 1.1.2(b)). It is the orthorhombic phase of $YBa_2Cu_3O_{7-\delta}$ which can superconduct. This phase has lattice constants $a = 3.80$ Å, $b = 3.86$ Å and $c = 11.55$ Å. The a and b axes alternate across an anti-phase boundary which runs parallel to the [110] direction. Because the magnitudes of the a and b dimensions are so similar, with $2(b-a)/(a+b) \approx 0.01$, the a and b occasionally interchange directions during sample preparation, crystal growth, or cooling down through the tetragonal to orthorhombic transition. This phenomenon is called twinning, and the twin-planes are the (110) planes. Because most single crystal and thin-film samples of $YBa_2Cu_3O_{7-\delta}$ are extensively

twinned, they act as though they were symmetric about the c-axis; the anisotropy on the *ab*-plane is mostly averaged out. However, the material is very anisotropic in the c-direction; the effective mass anisotropy parameter $\gamma = (m_c/m_{ab})^{1/2}$ has been measured to have a value between 5 and 10 [14][15], where m_c and m_{ab} are the Ginzburg-Landau superconducting effective masses for pair motion along the c-direction and in the *ab*-plane respectively. The lower critical field along the *ab*-plane $H_{c1,\parallel}(T=0)$ has been measured to be 250 ± 20 Oe, while along the c-axis $H_{c1,\perp}(T=0) = 850 \pm 40$ Oe [16], while the upper critical fields have been *derived* to be $H_{c2,\parallel}(T=0) = 140$ T and $H_{c2,\perp}(T=0) = 29$ T [17]. The penetration depth $\lambda_{ab}(T=0)$ has been measured to be 1400 Å [18], while $\xi_{\parallel}(T=0) \approx 34$ Å and $\xi_{\perp}(T=0) \sim 3.8 - 7$ Å [19][17], where $\xi_{\parallel}$ and $\xi_{\perp}$ are the *ab*-plane and c-direction coherence lengths respectively, so the material is strongly type-II, with $\kappa \equiv \lambda/\xi \gg 1$.

Bi-Sr-Ca-Cu-O. Among the copper-oxide superconductors, $Bi_2Sr_2CaCu_2O_8$ is one with the most anisotropic in properties. For instance, using torque magnetometry, Farrell *et al* [20] found the effective mass anisotropy parameter γ to have a value of 55 ± 5, substantially larger than that of Y-Ba-Cu-O. That means the effective mass in the c-direction is about 3×10^3 times as large as that in the *ab*-plane. The coherence lengths also show larger anisotropy than Y-Ba-Cu-O, with $\xi_{\parallel} \approx 42$ Å and $\xi_{\perp} \approx 1$ Å [21].

The compound $Bi_2Sr_2CaCu_2O_8$ has a critical temperature of about 85 K. It crystallizes in tetragonal structure, with two formula units per unit cell and lattice parameters $a = b = 3.817$ Å and $c = 30.6$ Å [12]. The crystallographic structure is shown in Figure 1.1.3. Because of the tetragonal structure, there is no twinning in the single crystals or thin-films. This absence of twin boundaries in $Bi_2Sr_2CaCu_2O_8$ naturally aroused interests in the flux-pinning capability of the compound, since in the case of Y-Ba-Cu-O, the pinning force in the crystals is often ascribed to the twin boundaries. R.B. van Dover *et al*, [22] using I-V measurements, find that at above ~ 35 K, the effective flux-pinning force in $Bi_2Sr_2CaCu_2O_8$, is extremely small. Because of this, the critical

current density at above 35 K also becomes very small, due to the dissipation caused by the flux line motion; below $\sim$ 20 K, the critical current density is high and only weakly field-dependent. These authors attribute this behavior to two possible reasons. The first one is that at above 35 K the pinning force itself may be small, leading to a high flux-creep rate; the resulting movement of the flux vortices due to the "creep" causes dissipation and thus reduces the critical current density. The second possible reason is that the Abrikosov flux lattice may melt at above 35 K, causing its shear modulus to vanish, so that the few pinned vortices do not prevent motion of the rest of the vortices. In Chapter 4, we will find that in our $Bi_2Sr_2CaCu_2O_8$ single crystal sample, the critical current density at 77 K, measured by ac susceptibility experiments, is extremely small, in agreement with the picture of a small effective pinning force.

1.2 Granularity of High-T_c Superconductors

Soon after the discovery of the first high-temperature copper-oxide superconductors, the materials were suspected to have a granular nature. For instance, transport critical current densities of ceramic Y-Ba-Cu-O were found to drop by an order of magnitude when a small dc magnetic field of about 25 Oe was applied to the sample [23]. Many authors thought the fact that the critical current densities were unusually sensitive to small magnetic fields was because they were the averaged critical current densities of the Josephson weak links which existed inside the ceramics and possibly formed a 3-dimensional matrix. In this matrix is, presumably, formed an array of highly superconducting grains coupled together through the weak links [2][24]. In support of this, Esteve *et al* [25] found that the I-V curves of a nonsuperconducting aluminum tip and a superconducting ceramic $La_{1.85}Sr_{0.15}CuO_4$ showed the characteristics of a Josephson junction, including Shapiro steps upon microwave irradiation, thus proving the existence of Josephson junctions inside their ceramic $La_{1.85}Sr_{0.15}CuO_4$ sample.

From susceptibility and magnetic-moment measurements on ceramic and powder samples of $(La_{1-x}Ba_x)_2CuO_{4-y}$, Müller, Takashige and Bednorz [26] found a quasi de Almeida-Thouless line [27], $D(H,T^*)$, in the H-T space, separating the magnetically reversible and irreversible regimes, and obeying empirically the expression $H = 1.17\ [1 - T^*(H)/T(0)]^{3/2}$, where $T(0) = 23$ K is the critical temperature of the samples and H is measured in teslas. Since from a theoretical point of view, de Almeida and Thouless first derived the line separating ergodic and nonergodic regions from the spin-glass model, Müller, Takashige and Bednorz concluded the existence of superconductive glass state [28] in their $(La_{1-x}Ba_x)_2CuO_{4-y}$ samples. Moreover, by equating H_{c1} of the superconductive glass to $\Phi_0/2S$, where Φ_0 is the flux quantum, and finding that the homogeneous superconducting area S was smaller than the grain size, they concluded that the superconductive glass state existed *within* the metallographically observed grain size.

This conclusion sparked debates about whether the weak links in the ceramics are located between the metallographically observed grains or within them, or both (see, for example, Ref [29]). While it is not too difficult conceptually to visualize the existence of weak links between individual highly-superconducting grains pressed and sintered together into a ceramic, *intra*granular weak links are not as simple to explain. One reason often cited for the possible existence of intra-granular weak links is the smallness of the coherence lengths in the oxide superconductors. The short coherence lengths cause weakening of the pair potential at surfaces, interfaces, and possibly even at twin boundaries between domains of different crystalline orientations.

Some microwave experiments seem to indicate that Josephson junctions do indeed exist at twin boundaries. For example, narrow absorption lines periodic in static magnetic field were observed by Blazey *et al* [30][31][32][10] at microwave frequencies in thin single crystals of micro-twinned Y-Ba-Cu-O. The static field in these experiments is in the *ab*-plane, while the microwave field is normal to the plane. The periodic absorption

lines are believed to be due to microwave-current induced fluxon nucleation in rf-SQUID structures which naturally existed in the Y-Ba-Cu-O single crystals, probably with the twin-boundaries acting as Josephson junctions.

In short, according to the superconductive glass picture, Josephson weak links that exist in the high-temperature superconductors, either inter- or intragranular, together with the relatively high temperatures at which the oxide superconductors are usually subject to in experiments, account for the glassy or granular behavior of the materials.

Later, Yeshurun and Malozemoff [33] observed the quasi de Almeida-Thouless line in single crystals of Y-Ba-Cu-O and thus showed that the line is not only characteristic of ceramic samples. Also, instead of invoking the more novel superconductive glass model, they argued that the conventional flux-pinning and flux-creep models [34][35] were able to explain the "glassy" behavior of the crystals, including the empirical fact that the quasi de Almeida-Thouless irreversibility line obeys $(1 - T/T_c) \propto H^{2/3}$, using simple scaling arguments. They cited the direct flux-line decoration in crystals and the observation of conventional hysteresis loops interpretable in the Bean critical-state model [36] as supports for the conventional theories.

However, even in this "conventional" picture, the smallness of the coherence lengths and the high temperatures also play an important role. This makes one wonder if the two pictures — superconducting glass and conventional flux-pinning/flux-creep — are really mutually exclusive. Later in Chapter 4 of this thesis, we will show experimental evidence that, at least for the intergranular medium of ceramic Y-Ba-Cu-O, both the superconducting glass model and the critical state model are valid and applicable to describe different aspects of the same system.

In Chapter 2, several models regarding the nonlinear nature of the electrodynamical behavior of granular and type-II superconductors are developed. These models predict novel properties including extensive harmonic generation by an applied ac magnetic

field, with dips in the harmonic power as a function of an applied dc magnetic field. The models also provide methods to quantitatively deduce the critical current densities, both intrinsic and intergranular in the case of ceramic samples, and to tell experimentally if there is more than one supercurrent component in a sample. Details about the actual model calculations are given in Appendices A and B.

Details about the experimental procedures are presented in Chapter 3. The samples relevant to the data presented in this thesis are listed and described. Actual data acquisition programs using an AT-compatible computer will be presented in Appendix C.

Experimental results, analyses, and comparison to models are presented in Chapter 4. Extensive harmonic generation by the high-T_c compounds is observed. The dependence of the harmonic power on dc and ac magnetic fields, and temperature, are measured and compared to model calculations. The question raised earlier about the novel superconducting glass model versus conventional critical state model will also be addressed for the ceramic Y-Ba-Cu-O. Experimental evidence supporting both models simultaneously will be presented. Evidence proving the coexistence of inter- and intragranular supercurrents in a single ceramic sample are given, together with quantitative estimates of their respective critical current densities. The separate transition temperatures for the inter- and intragranular superconductivity in a ceramic sample are measured and presented. A summary is given in Chapter 5.

1.3 Figure Captions and Figures of Chapter 1

Figure 1.1.1. Perovskite cubic unit cell showing titanium on the apices and oxygen in the edge-centered positions. Barium, which is in the body center, is not shown. [12]

Figure 1.1.2. Sketches of the (a) tetragonal and (b) orthorhombic yttrium-barium-copper oxide unit cells. Oxygens are randomly dispersed over the basal plane sites in the tetragonal structure. Thermal vibration ellipsoids are shown for the atoms. [12]

Figure 1.1.3. The structure of $Bi_2Sr_2CaCu_2O_8$, sometimes called Bi(2212). [11]

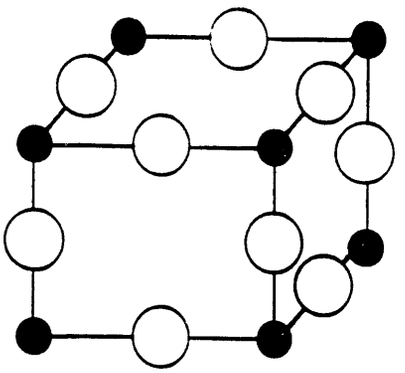


Figure 1.1.1

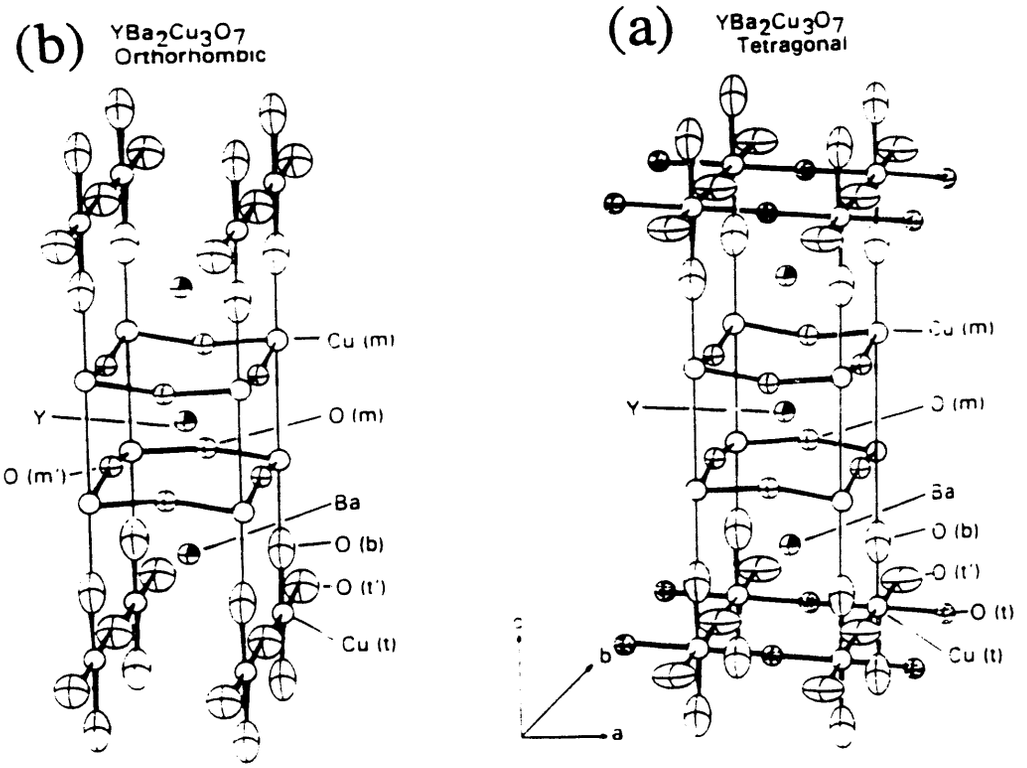


Figure 1.1.2

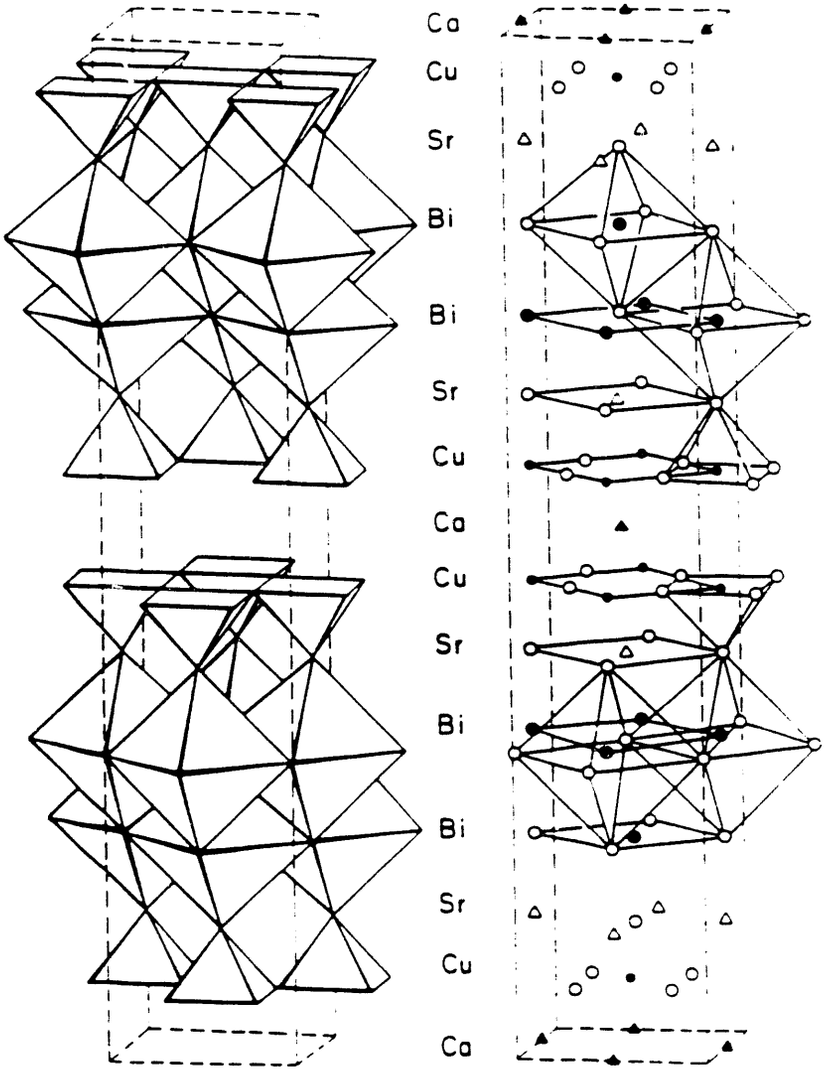


Figure 1.1.3

Chapter 2 Models

In this chapter we introduce two classes of models by which we shall explain our experimental data in Chapter 4. The first class of models, presented in Section 2.1, is designed to explain the nonlinear electrodynamical behavior of powdered high-T_c samples. It assumes that each grain inside the sample behaves as a superconducting loop. Due to flux quantization of the loops, interesting properties of the powdered samples, which is modeled to be an ensemble of loops with different areas, are predicted.

The second class of model, presented in Section 2.2, is based on and modified from the critical-state model designed originally by C. P. Bean [36] back in the 1960's to explain flux trapping and dc magnetization hysteresis in low-temperature type-II superconductors. The model is generalized in this section and will be used later in Chapter 4 to explain data taken on a bulk cylindrical ceramic $YBa_2Cu_3O_{7-\delta}$ sample, a $YBa_2Cu_3O_{7-\delta}$ thin-film and a $Bi_2Sr_2CaCu_2O_8$ single crystal.

2.1 Superconducting Loop Models

In this section, a novel class of models is presented which predicts interesting nonlinear electrodynamical behavior from an ensemble of superconducting current loops with a wide distribution of loop areas. These models are later used to explain experimental data of high harmonic generation taken on powdered $YBa_2Cu_3O_{7-\delta}$ sample.

The idea of the models is shown schematically in Figure 2.1.1. The metallographically observed grains in a powdered superconducting sample, Figure 2.1.1(a), are assumed to contain Josephson weak links. Such weak links may be due to twin boundaries, cracks in the crystal structures introduced during the various grinding processes, or that the grains themselves are composed of subgrains which are in electrical contact with one another through weak links, such as shown in Figure 2.1.1(c). In this class of models, we concentrate on the junction current density j_J which flows through these "intragranular"

junctions, which for low fields ($\leq$ 50 Oe) will usually dominate the intrinsic currents. We assume that this intragranular junction current j_J can be represented prototypically by an rf-SQUID-like superconducting current loop, Figure 2.1.1(d); the Josephson junction in the prototype rf-SQUID represents the weakest link among the possibly many links in Figure 2.1.1(c).

In the "zero-order model," a prototype j_J loop in the ensemble is assumed to behave as a lossless rf-SQUID; the normal current component of the two-fluid model is ignored. The current-phase relationship of the Josephson junction is assumed to be purely sinusoidal. Also ignored is the self-field generated by the loop-current which tends to reduce the flux through the loop.

In the "loop model," the possibility and effects of a non-sinusoidal, periodic current-phase relationship are considered. However, loss is still ignored in this model.

In the "first-order model," loss is introduced into the model by adding a shunt-resistance across the Josephson junction in the prototype rf-SQUID-like loop. Hysteretic loss is also made possible by introducing a self-inductance to the loop. This model may account for some of the loss mechanisms which manifest themselves in the data.

2.1.1 Zero-order model

A superconducting powdered sample is considered to be subjected to parallel and uniform dc and ac fields, the total applied field being $H = H_{dc} + H_1 \sin(\omega t)$. The sample is assumed to have no electrical contacts but is surrounded by a solenoid; all measurements are made from the voltage induced into this "receiver" coil. Each current loop j_J of the powdered sample is assumed to behave as an idealized superconducting current loop, with a weak link in its current path, Figure 2.1.1(d). The weak link may be a Josephson tunnel junction or a point contact. For low fields $H < H_{c1}$ of the intrinsic material, the situation will be modeled by an ensemble of superconducting paths intersected by weak links, the specific prototype being a thin ring-shaped loop of

area S_0 in series with a junction of area s_0, such that the flux due to the applied field is $S_0 H \cos\theta$ and $s_0 H \cos\phi$, respectively. Note that this geometry is similar to that used to model rf superconducting quantum interference devices (SQUID's). In the zero-order model we neglect the flux due to the loop current itself, but reconsider it below in the first order model. The electromagnetic properties of the sample are then predicted by taking suitable averages over a distribution of areas S_0 , s_0 and orientations θ, ϕ. Let us define the dimensionless quantities

$$h_{dc} \equiv \frac{2\pi S_0 H_{dc} \cos\theta}{\Phi_0}, \qquad h_1 \equiv \frac{2\pi S_0 H_1 \cos\theta}{\Phi_0}, \qquad \eta \equiv \frac{\pi s_0 H \cos\phi}{\Phi_0}, \tag{2.1.1}$$

where Φ_0 is the flux quantum and $h_{dc} / 2\pi$ is just the number of flux quanta in the loop due to H_{dc}, etc. The applied field induces current in the prototype loop, which, for a tunnel junction is given by the Josephson current-phase relation

$$I(t) = I_c \left(\frac{\sin\eta}{\eta} \right) \sin\left(\gamma(t)\right) , \tag{2.1.2}$$

where $\gamma(t) \equiv h_{dc} + h_1 \sin\omega t$, and I_c is the junction critical current. We assume that the junction area s_0 is sufficiently small that the diffraction term $[\sin\eta/\eta] \approx 1$, and consider only the Fourier components of $\sin\gamma$, arising from flux quantization of the loop:

$$\begin{aligned} I_{l0} &= I_c J_0(h_1) \sin(h_{dc}) , \\ I_{ln} &= \begin{cases} 2 I_c J_n(h_1) \sin(h_{dc}) \cos(n\omega t), & n \text{ even}, \\ 2 I_c J_n(h_1) \cos(h_{dc}) \sin(n\omega t), & n \text{ odd}, \end{cases} \end{aligned} \tag{2.1.3}$$

where $J_n(h_1)$ is the Bessel function of integer order n.

We assume that each superconducting loop with area S_0 induces a receiver coil harmonic signal voltage $v_n(t)$ proportional to $S_0 \cos\theta \, dI_{ln}/dt$. If the sample were composed of only one loop the signal power $P(n\omega) \propto v_n^2$ for all harmonics would be periodic in H_{dc} due to flux quantization, with period $\Delta h_{dc} = \pi$, i.e., $\Delta H_{dc} = \Phi_0 / (2 S_0 \cos\theta)$ between dips, corresponding to the period of $\cos^2 h_{dc}$ or $\sin^2 h_{dc}$. We characterize the ensemble of current loops by a uniform distribution of orientation angles

and an area distribution function $F(A)$, with $A \equiv S/S_0$. All the loops are assumed to be coherently driven, so that the total signal voltage $V_n(t)$ at some harmonic $n\omega$ can be represented by the algebraic sum of all $v_n(t)$. The sample-average signal amplitude $\langle V_n \rangle$ is computed by the expression,

$$\langle V_n \rangle = \frac{n\omega}{G} \int_{A=\delta}^{\infty} dA \int_{\epsilon=0}^{1} A\epsilon\, J_n(Ah_1) \cos(Ah_{dc})\, F(A)\, d\epsilon, \text{ odd } n; \qquad (2.1.4a)$$

$$\langle V_n \rangle = \frac{n\omega}{G} \int_{A=\delta}^{\infty} dA \int_{\epsilon=0}^{1} A\epsilon\, J_n(Ah_1) \sin(Ah_{dc})\, F(A)\, d\epsilon, \text{ even } n; \qquad (2.1.4b)$$

where $\epsilon \equiv \cos\theta$ and G is the normalizing factor $\int\int F(A) \sin\theta\, dA\, d\theta$. For later reference we also include the average sample magnetization $\langle M_z \rangle$ computed under the same averaging assumptions, using the dc current term I_{l0} in Eqn. (2.1.3)

$$\langle M_z \rangle = \frac{I_c}{GV_{tot}c} \int_{A=\delta}^{\infty} dA \int_{\epsilon=0}^{1} \epsilon A \sin(Ah_{dc})\, F(A)\, d\epsilon\,, \qquad (2.1.5)$$

where V_{tot} is the total volume of the sample.

To examine the effects of averaging on $P(n\omega)$ vs h_{dc}, we take as an *example* a Gaussian distribution function for loop areas

$$F(A) = \exp\left\{ -\frac{(A-1)^2}{2\sigma^2} \right\} \qquad (2.1.6)$$

peaked at $A = 1$, or $S = S_0$, with standard deviation σ. First, to represent a single loop we take $\sigma = 0$, and $\cos\theta = 1$ in Eqn. (2.1.4) and compute $P(n\omega)$, plotted in Figure 2.1.2(a), which shows the expected periodicity $\triangle h_{dc} = \pi$; this plot is valid for all values of ac field amplitude h_1 and odd n. Next, for standard deviation $\sigma = 2$, $h_1 = 5$, and $n = 1$ we compute $P(n\omega)$, plotted in Figure 2.1.2(b). We see that the periodicity is "averaged out" for a distribution of areas and orientations. However, for large values of n the result is different. For $\sigma = 2$. $h_1 = 5$, and $n = 15$, we compute and plot $P(15\omega)$ in Figure 2.1.2(c), finding deep and almost periodic dips, with an average dip spacing $\overline{\triangle h_{dc}} = 1.03$. If we omit the averaging over θ in Eqn. (2.1.4) the plot is essentially

the same as Figure 2.1.2(c), with $\overline{\Delta h_{dc}}$ smaller by 1.5 %. For increased σ, the plots are very similar, with decreased $\overline{\Delta h_{dc}}$; the pattern converges for $\sigma \geq 2$. Essentially, the same behavior is found for other values of *n*, with $\overline{\Delta h_{dc}} \propto n^{-1}$ for $n \gg 1$. If we include the $[\sin\eta \,/\, \eta]$ term in Eqn. (2.1.2), the computed shapes of $P(n\omega)$ for small *n* are modified to an extent depending on the distributions of *S* and *s*. However, for large *n*, the shapes are not sensitive to the details of either *S* and *s* distributions, as long as they are monotonically decreasing at large areas.

The principal result of the rf-SQUID model is that this model of powdered high-temperature superconductors, even with a broad distribution of areas and grain orientations, predicts sharp and almost periodic dips in the harmonic power as the dc field is varied, perhaps giving evidence of an effective flux quantization arising from the loop of the model. One would have naively expected the periodic flux quantization of the individual loops to be generally averaged out by the wide distribution of loop areas, this is not so for high harmonics. Other distribution functions $F(A)$ also yield sharp dips in $P(n\omega)$ vs. h_{dc} for large *n*. At this point it is instructive to observe that the distribution function $F(A)$, *assumed* to be a Gaussian for illustrative purposes in Eqn. (2.1.6), can instead be experimentally determined by a second independent measurement: the dc magnetization $M(H)$; this quantity is predicted by the model in Eqn. (2.1.5). It is generally found by many observers (see, for example, [23][37]) that $M(H)$ in low fields ($H \leq 25$ Oe) shows initially a linear behavior and then saturates. We *approximate* this behavior by the simple analytic expression

$$\frac{M(H_{dc})}{M_0} \approx \tanh\left(\frac{H_{dc}}{H'}\right), \tag{2.1.7}$$

where H' is the field where saturation begins. In order to get an analytical Fourier sine transform from this dc magnetization expression, Eqn. (2.1.7) is further approximated as

$$\frac{M(H_{dc})}{M_0} \approx \frac{\sinh(\zeta H_{dc}/H')}{\cosh(H_{dc}/H')}, \tag{2.1.8}$$

with ζ smaller than but close to one. If we equate this "empirical" result with Eqn. (2.1.5), the function $F(A)$, with $\zeta < 1$, is determined [38]; in the limit of $\zeta \rightarrow 1_-$,

$$F(A) = \frac{\sinh(A\pi/2)}{A\,[\cosh(A\pi) - 1]} . \tag{2.1.9}$$

This distribution function will be used in Chapter 4 to fit the experimental data.

2.1.2 Loop model

We now explore the possibility that the current-flux relation of the individual prototype current loop is not sinusoidal, as in Eqn. (2.1.2), but still periodic with period Φ_0 . There are several conceivable cases in which this occurs: (i) The current-phase relation of the weak links may deviate from the pure sinusoidal form of Eq. (2.1.2), which was derived by Josephson for the case of a weakly coupled tunnel junction; (ii) the prototype loop has a *large* number of identical junctions, and the change in the loop current is then controlled by the change in phase-winding number of the loop rather than by the current-phase relation of individual junctions; (iii) screening by the loop current effectively gives a skewed periodic current-*applied flux* relation, as in the case of rf-SQUIDs; and (iv) there may be current loops which simply are superconducting without any junction or weak link in their paths. We now consider this last special case, although the results should be applicable to the others. Fluxoid quantization in a loop requires that

$$\int_S \mathbf{H} \cdot d\mathbf{S} + \left(\frac{m^* c}{2e}\right) \int_l \mathbf{v} \cdot d\mathbf{l} = n\Phi_0 , \quad n = 0, 1, 2, \ldots , \tag{2.1.10}$$

which, for a thin ring of radius R, yields the velocity $\mathbf{v}$ of the superconducting electrons and, hence, the current density $I_l \sim v = \hbar\,(n - \Phi/\Phi_0)\,/\,(m^*\,R)$, where $\Phi = H\pi R^2$ is the applied flux through the ring. The kinetic energy is proportional to $(n - \Phi/\Phi_0)^2$. As the flux Φ/Φ_0 is increased we allow n to switch from $n = 0$ to 1, etc., maintaining the system in a minimum kinetic energy state. The current I_l is then a sawtooth function

of Φ/Φ_0 which we write as the Fourier series

$$I_l = \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{m} \sin\left(\frac{m\,2\pi\Phi}{\Phi_0}\right) , \tag{2.1.11}$$

with $2\pi\Phi/\Phi_0$ to be identified with $h_{dc} + h_1 \sin(\omega t)$ in Eqn. (2.1.1). Following the same procedure used to obtain Eqn. (2.1.4), we use Eqn. (2.1.11) to find, for odd n, the sample-average signal voltage components

$$\langle V_n \rangle = \frac{n\omega}{G} \sum_{m=1}^{\infty} \frac{(-1)^{m+1}}{e^{m-1}} \int_0^{\infty} dA \int_0^{\pi/2} A \cos\theta\, J_n(mAh_1) \times \cos(mAh_{dc})\, F(A) \sin\theta\, d\theta , \tag{2.1.12}$$

where for convergence we have replaced $1/m$ in the summation in Eqn. (2.1.12) by $\exp(-m+1)$ to round off the high harmonics of an otherwise infinitely sharp sawtooth. From Eqn. (2.1.4), one can see that the zero-order model is merely the first term of Eqn. (2.1.12). Plots of $P(n\omega)$ vs. h_{dc}, computed from Eqn. (2.1.12) are found to be quite similar to the zero-order model; however, at small values of h_1 the loop model predicts additional structures. Shown in Figure 2.1.2(d) is $P(2\omega)$ vs. h_{dc} computed from Eqn. (2.1.12) for $h_1 = 0.5$. This is to be compared to $P(2\omega)$ for the zero-order model, Figure 2.1.2(e), computed for $h_1 = 0.5$ and the same $F(A)$ as Figure 2.1.2(d).

2.1.3 First-order model

Note that the models in Section 2.1.1 and 2.1.2 do not contain any source of dissipation. For $n = 1$ the model signal voltage ($\sim \cos\omega t$) is just in phase with the leakage signal $dH_{ac}/dt \sim \cos\omega t$; these models do not yield an imaginary part of the complex susceptibility, which is neither realistic nor in agreement with the data.

So far we have made the assumption that the self-induced flux due to the current circulating in the loop could be neglected. We have also neglected the resistive current flowing in the loop. However, these assumptions ignore dissipation in the sample which can be caused by either the resistive current or bulk-pinning hysteresis. A result of these assumptions is that Eqns. (2.1.3) only give the inductive components in the receiver

coil signal. As an attempt to describe the system more realistically, we generalize the zero-order model by assigning a self-inductance to the loop and adding a resistance R in shunt with the junction. The loop current is then given by $I(t) = I_c \sin\gamma(t) + V/R$ where $V = (\hbar/2e)\, d\gamma/dt$ and $\gamma = h_{dc} + h_1 \sin(\omega t) - 2\pi LI/\Phi_0$. Combining these expressions one obtains

$$\frac{1}{\omega}\frac{dI_1}{dt} = \frac{1}{\kappa L_0}\{\sin[h_{dc} + h_1 \sin(\omega t) - L_0 I_1] - I_1\} + \left(\frac{h_1}{L_0}\right)\cos(\omega t)\ , \tag{2.1.13}$$

where $I_1 \equiv I/I_c$, $\kappa \equiv \hbar\omega / 2eRI_c$, $L_0 \equiv 2\pi LI_c / \Phi_0$. For given values of parameters h_{dc}, h_1, κ, and L_0 and loop area $S = AS_0$, Eqn. (2.1.13) is numerically iterated to yield dI_1/dt. This quantity is averaged over a Gaussian distribution of areas, Eqn. (2.1.6), with L_0 assumed to vary as $A^{1/2}$, to obtain $\langle V(t)\rangle_A$, then the spectral components of which are computed using a fast Fourier transform algorithm, yielding real and imaginary components $V_{real}(n\omega)$ and $V_{imag}(n\omega)$. The corresponding power $P(n\omega)$ is plotted versus h_{dc} in Figure 2.1.2(f) for $n = 15$, $h_1 = 5$. Although there is a clear correspondence with Figure 2.1.2(c), one sees that now the inductive and dissipative terms have a different dependence on h_{dc} so that the dips have a more complex pattern.

2.2 Generalized Critical State Model

In this section, we review the critical state model which was proposed originally by C. P. Bean [36] to explain flux trapping and dc magnetization hysteresis in conventional type-II superconductors. We also introduce a generalized version of the model which will be more suitable to explain the experimental data on high-T_c superconductors.

Let us suppose a type-II superconductor penetrated by a magnetic field $H > H_{c1}$. When this magnetic field is first applied, it induces a bulk screening current in the superconductor. This current will be given by

$$\mathbf{J} = \left(\frac{c}{4\pi}\right)\nabla \times \mathbf{H} \tag{2.2.1}$$

and it will try to prevent the flux lines nucleated at the surface from moving into the superconductor. The magnetic field will penetrate in the form of Abrikosov flux lines, ie. "vortices" or "fluxons"; the flux density in the sample is clearly not spatially uniform because of the current. The magnetic energy per unit volume, or the magnetic pressure exerted by the flux lines on one another, is $HB/8\pi$. In the case when the distances between flux lines are small compared to the penetration depth, λ, this magnetic energy $HB/8\pi$ is equal to the interaction free energy density of the flux lines. This would mean that the force on the flux lines per unit volume would be given by the gradient of the interaction free energy density $\nabla F_{int} = -\mathbf{B} \times \nabla \times \mathbf{H}/4\pi = \mathbf{J} \times \mathbf{B}/c$, ie., the Lorentz force; the force per flux line is then $\mathbf{J} \times \mathbf{\Phi}_0/c$ per unit length, where $\mathbf{\Phi}_0$ has a magnitude of the flux quantum $hc/2e$ and the same direction as that of the flux line. Throughout this thesis, we will assume a linear relation between B and H: $\mu_{eff} = B/H$. For an intrinsically nonmagnetic homogeneous material, $\mu_{eff} = 1$.

The interaction between flux lines is relatively long-ranged. Because of this, local perturbations of the line density are very unfavorable energetically; simply putting in locally one extra flux line costs an energy of the order of $H\Phi_0$ per unit length, much greater than the energy available from any reasonable pinning centers [34]. Thus on a length scale smaller than the penetration depth, the density of the flux lines is well-defined. In the case of a granular superconductor, it will then be reasonable that if the intergranular medium's penetration depth λ_J is greater than the typical grain dimension, a, then the local flux line density, or magnetic induction, $\overline{B(\mathbf{r})}$ should be well-defined in spite of the granularity and the intergranular medium can be treated as a continuum type-II superconductor with an effective permeability

$$\mu_{eff} \equiv \frac{\overline{B(\mathbf{r})}}{\overline{H(\mathbf{r})}} \ . \tag{2.2.2}$$

Here the overline stands for an average over a volume scale larger than a^3, but much smaller than the sample volume. This effective permeability reflects the ratio of the

volume of the intergranular weak link region to the total volume of the sample. The intergranular weak link region's volume is dependent on temperature because it includes the London penetration depth of the highly superconducting grains. Thus for a granular superconductor [39],

$$\mu_{eff}(T) = f_n + f_s \left[1 - F\left(\frac{R_g}{\lambda_g(T)}\right) \right], \tag{2.2.3}$$

where f_n and f_s are the nonsuperconducting (including voids) and superconducting volume fractions, respectively, and $f_n + f_s = 1$. R_g and λ_g are the grains' (averaged) radius and London penetration depth respectively, and $F(R_g/\lambda_g)$ is the factor by which the magnetic flux peneuration suppresses a grain's magnetization below that expected for complete Meissner-state flux exclusion.

If the flux lines are not pinned, the Lorentz force, $J\Phi_0 / c$, acting on them will cause them to flow. In the steady state of the flux flow, the Lorentz force on the flux lines will be balanced by the viscous drag exerted by the material of the superconductor on the vortices, and a constant flow velocity will be attained. The work done against the viscosity will appear as an emf against the current. This is the origin of the flux-flow resistivity in type-II superconductors.

If, however, the flux lines are somehow immobilized in the superconductor, zero resistance will be allowed. Defects and other pinning centers in the material, for instance, may be able to trap the flux lines and thus fix them in their positions. When a transport current (as distinct from the supercurrents around the flux vortices) is flowing in the superconductor, the resulting Lorentz force will try to depin the flux lines from their pinning sites. Thus the Lorentz force density being smaller than the pinning force density α, $JB / c < \alpha$, is the requirement for dissipationless current flow in the type-II superconductors. Note that for simplicity, we have ignored all possible thermal effects in our discussion such as thermally activated flux creep, which is still an unresolved issue.

In the critical state model, it is assumed that when the external magnetic field applied on the type-II superconductor is changed, a current will be induced in such a way as to oppose a corresponding change in the density of the magnetic flux through the superconductor. For instance, if the superconductor has been cooled in zero magnetic field ("zero-field-cooled") and the external field is subsequently increased from zero to a certain positive value larger than the lower critical field H_{c1}, current will be induced to reduce the amount of flux vortices, which are nucleated at the surface, penetrating into the material. On the other hand, if flux vortices already exist inside the superconductor due to previous application of magnetic field, reducing the external field, say to zero, will not cause all the penetrated flux to exit the superconductor, because current will be induced to prevent some of the flux from getting out.

Another assumption of the critical state model is that the current density induced by a changing field will be equal to the (local) critical current density of the material. That is to say, the current is induced to such a value that the Lorentz force density on the flux lines is equal to the flux pinning force density α:

$$J_c B / c = \alpha . \tag{2.2.4}$$

This value of current density is the maximum value at which a dissipationless current flow is still allowed; beyond that is the resistive flux-flow regime. Hence, this maximum supercurrent density J_c is called the critical current density. Eqn. (2.2.1) can then be rewritten as

$$\nabla \times \mathbf{H} = \left(\frac{4\pi}{c}\right) \mathbf{J}_c . \tag{2.2.5}$$

Because of this finite value of critical current density, the type-II superconductor has a limited ability to screen or trap flux. For instance, in case of an increasing external field applied on a zero-field-cooled sample, it can only screen the penetrating flux up to a certain depth into the material; the flux density will, in this case, decrease

from a maximum value at the surface, to zero after penetrating a certain depth. When the external field is further increased, the supercurrent already flowing in the surface layer, being at its critical value, cannot prevent more flux lines from penetrating further. However, supercurrent will then be induced in a region deeper into the sample which has previously been unexposed to the flux vortices, and will stop the penetration of the flux vortices at this deeper level. At some point in increasing the external field, the penetrated flux front will reach the center of the sample, and supercurrent will begin to flow throughout the sample. The external field at this point is of significance in experiments presented later and is denoted by H^*. As will be mentioned in more detail later, in an ac susceptibility experiment, the dissipative component χ'' of the complex susceptibility $\tilde{\chi} \equiv \chi' - i\chi''$ will attain a peak at H^* when measured as a function of ac field amplitude H_1. For convenience, we will refer to H^* as the "penetration field."

In the original Bean version of the critical state model, the critical current density is assumed to be independent of the magnetic field. For a cylindrical geometry, Eqn. (2.2.5) can then be written as

$$\frac{dH}{dr} = \pm \frac{4\pi}{c} J_c = constant\,, \tag{2.2.6}$$

where the sign is determined by the direction of the supercurrent flow in the local region, or, in other words, by whether the local current is trying to screen out or to trap in flux vortices. Because J_c is taken to be constant, both the magnetic field H and the flux density B are linear as a function of the radius. In Figure 2.2.1(a), the field profile is plotted across the sample for several values of increasing external field. Note that when $H_{ext} = H^*$, the flux front reaches the center. In Figure 2.2.1(b), the external field is assumed to have been reduced back to zero from a non-zero maximum field value H_{max}. Flux is trapped in the sample, indicated by the shaded area. Flux trapping such as this would account for hysteretic behavior in dc magnetization measurements of type-II superconductors.

The critical state model was later extended from the Bean version by Kim and Anderson to cases in which the critical current density is dependent on the local magnetic field. They, instead of taking J_c as a constant, assume a constant flux pinning force density, α, in the sample. Thus the critical current density can be written as $J_c(H) = \alpha c / (\mu_{eff} |H|)$. To prevent the singularity of J_c at $H = 0$. a phenomenological parameter H_0 is added to the J_c expression:

$$J_c(H) = \frac{\alpha c}{\mu_{eff} (|H| + H_0)} . \tag{2.2.7}$$

The critical state equation in the cylindrical geometry then becomes

$$\frac{dH(r)}{dr} = \pm \frac{4\pi\alpha}{\mu_{eff} (|H(r)| + H_0)} . \tag{2.2.8}$$

For a conventional type-II superconductor, Anderson [34] estimates that the positive parameter H_0 is of the order of $\Phi_0 / (\mu_{eff} \lambda^2)$, and its appearance in Eqn. (2.2.7) is due to the discreteness of quantized flux; any movement of flux must involve a minimum value of Φ_0. When applied to the intergranular medium of a granular superconductor, presumably containing Josephson weak links between the grains, H_0 is estimated to be of the order of $\Phi_0 / (\mu_{eff} s_0)$, where s_0 is a characteristic area for the weak links [40].

While Eqns. (2.2.7) and (2.2.8) have been rather satisfactory in explaining dc magnetization measurements on conventional type-II superconductors, there has been evidence that Eqn. (2.2.7) may not be an accurate enough description for the transport critical current density measured in ceramic high-temperature superconductors. For example, in Ref. [41], it is found that the critical current density J_c measured by transport experiment on a ceramic Y-Ba-Cu-O drops as $\sim H^{-2}$ upon the application of a dc magnetic field. In terms of Eqn. (2.2.7), it would seem that the flux pinning force density α is itself a function of magnetic field. In order to generalize the critical state model to accommodate this possibility of the H-dependence of α, and to include the Bean version, Kim-Anderson version and the empirical $J_c \sim H^{-2}$ relation with a

minimum of parameters, Eqn. (2.2.7) is generalized to [42][9][43]

$$J_c(H) = \frac{\alpha(H)\, c}{\mu_{eff}\,(|H| + H_0)} = \frac{\alpha' c}{(|H| + H_0)^{\beta}} , \tag{2.2.9}$$

with β an adjustable parameter, and where α' is taken to be a field-independent parameter. Eqn. (2.2.8) will then become

$$\frac{dH(r)}{dr} = \pm\frac{4\pi\alpha'}{(|H(r)| + H_0)^{\beta}} . \tag{2.2.10}$$

Note that when $\beta = 0$, Eqn. (2.2.10) will become the original Bean version, Eqn. (2.2.6); when $\beta = 1$, it will be the Kim-Anderson version, Eqn. (2.2.8). From Eqn. (2.2.10), the penetration field H^* can be derived as

$$H^* = \left[H_0^{\beta+1} + 4\pi\,(\beta+1)\,\alpha' R\right]^{\frac{1}{\beta+1}} - H_0 . \tag{2.2.11}$$

The full details of the calculation will be given in Appendix A. For the special case of the Bean-version,

$$H^* = \frac{4\pi}{c} J_c R ; \tag{2.2.12}$$

and for the Kim-Anderson version,

$$H^* = \left[H_0^2 + \frac{8\pi\,\alpha R}{\mu_{eff}}\right]^{\frac{1}{2}} - H_0 . \tag{2.2.13}$$

To illustrate the differences from the Bean version introduced by the H-dependence of J_c, field profiles for several values of external magnetic field are plotted for the Kim-Anderson model in Figure 2.2.2(a) and (b), and for the generalized model with $\beta = 1.8$, and $H_0 = 3$ in Figure 2.2.2(c) and (d).

According to the critical state model, when a type-II superconductor is subject to a purely sinusoidal ac magnetic field [44], say

$$H_{ac}(t) = H_1 \cos(\omega t) , \tag{2.2.14}$$

the magnetization of the sample will traverse a minor hysteresis loop. Because of this hysteretic behavior, the dependence of the total flux in the superconductor on time is nonsinusoidal. In Ref. [36], Bean predicts from Eqn. (2.2.6) generation of odd harmonics of the driving frequency in the voltage signal picked up by a secondary coil tightly wound on a cylindrical specimen with a constant J_c:

$$V(t) = V_1 \cos(\omega t - \gamma) + V_3 \cos(3\omega t) + V_5 \cos(5\omega t) + \ldots , \tag{2.2.15}$$

where

$$V_1 = 1.088\, V_3 \tag{2.2.16a}$$

$$V_3 = -\left[\frac{2\omega H_1^2 N R}{\pi J_c}\right] \times 10^{-8} \text{ volts} \tag{2.2.16b}$$

$$V_n = \left[\frac{5}{(n-2)(n+2)}\right] V_3 , \tag{2.2.16c}$$

where, in Bean's particular derivation, $H_1 < H^*$, and where N is the number of turns of the secondary coil and R is the radius of the sample. That the Bean version predicts only odd harmonic generation is due to the assumption that J_c is independent of H. For the Kim-Anderson version [35][45], Eqn. (2.2.8), or our generalized model, Eqn. (2.2.10), even harmonics are also predicted in addition to the odd ones when the dc magnetic field superposed on the ac field is non-zero.

In order to clearly explain how we are going to use the generalized critical state model to explain our experiments, let's for the moment take an experimental viewpoint. For a long cylindrical sample of radius R in a coil of N turns, in general one can write the pick-up voltage signal as

$$V(t) = -\frac{N\pi R^2}{c} \frac{d}{dt} \langle B(t) \rangle , \tag{2.2.17}$$

where $\langle B(t) \rangle$ is the flux density averaged over the cross-section area of the sample,

$$\langle B(t) \rangle \equiv \frac{\mu_{eff}}{\pi R^2} \int_0^R H(r,t)\, 2\pi r\, dr . \tag{2.2.18}$$

Note that the field value $H(r,t)$ itself already represents a spatial average of the microscopic magnetic field values over a volume greater than the average grain size but much smaller than the sample.

One can write Eqn. (2.2.18) as a Fourier expansion

$$\langle B(t)\rangle = \langle B_{dc}\rangle + \mu_{eff} H_1 \sum_{n=1}^{\infty} \left[\mu_n' \cos(n\omega t) + \mu_n'' \sin(n\omega t) \right] , \qquad (2.2.19)$$

where

$$\begin{aligned} \mu_n' &= \frac{\omega}{\pi \mu_{eff} H_1} \int_0^T \langle B(t)\rangle \cos(n\omega t)\, dt , \\ \mu_n'' &= \frac{\omega}{\pi \mu_{eff} H_1} \int_0^T \langle B(t)\rangle \sin(n\omega t)\, dt \end{aligned} \qquad (2.2.20)$$

are the components of the complex harmonic permeability; in most previous experiments only the $n = 1$ components were considered. From Eqns. (2.2.17) and (2.2.19), one obtains

$$V(t) = \mu_{eff} V_0 \sum_{n=1}^{\infty} n \left[\mu_n' \sin(n\omega t) - \mu_n'' \cos(n\omega t) \right] , \qquad (2.2.21)$$

and

$$V_0 = \frac{1}{c} N \pi R^2 H_1 \omega \qquad (2.2.22)$$

is the amplitude to the pick-up signal in the absence of the sample. We note that the complex permeability $\tilde{\mu}_n = \mu_n' - i\mu_n''$ is related to the complex susceptibility $\tilde{\chi}_n = \chi_n' - i\chi_n''$ by the relations $\tilde{\mu}_1 = 1 + 4\pi\tilde{\chi}_1$, and $\tilde{\mu}_n = 4\pi\tilde{\chi}_n$ for $n > 1$.

To compare the experimental data with the model, Eqn. (2.2.10) is solved analytically to find $H(r)$ as a function of the instantaneous applied magnetic field

$$H_{app}(t) = H(r = R, t) = H_{dc} + H_1 \cos(\omega t) , \qquad (2.2.23)$$

for $0 < r < R$, $0 < t < 2\pi/\omega$. From this we derive the analytical expressions for the magnetic flux in the sample

$$\Phi(H_{app}(t), t) = \pi R^2 \langle B(H_{app}(t), t)\rangle , \qquad (2.2.24)$$

and its time-derivative $d\Phi/dt$, where $\langle B(t)\rangle = \langle B(H_{app}(t), t)\rangle$ is defined by Eqn. (2.2.18). From a discrete time series (number of points = 2048) of $d\Phi/dt$ we used a fast Fourier transform [46][47] to compute the Fourier components μ_n' and μ_n'' in Eqns. (2.2.19) and (2.2.21), as functions of the experimentally known parameters (H_{dc}, H_1, R). However, because H^* is readily measured experimentally as the peak position of $\mu_1''(H_1)$, this leaves only two independent parameters, β and H_0 in Eqn. (2.2.10) to be selected by fitting the computed $\tilde{\mu}_n$ to the data. The analytical expressions relevant for the model calculations will be presented in Appendix A.

Details of the harmonic signals predicted by the generalized critical state model as a function of dc and ac magnetic fields will be given in Chapter 4 as fits to experimental data. In essence, the harmonics contain information on the details of the penetrated flux profile in the sample and, hence, the critical current dependence on magnetic field, $J_c(H)$.

As for the fundamental mode signal, the in-phase signal $V_0\mu_1'$ ("in-phase" defined to be $\sin(\omega t)$, which is the phase of the leakage signal induced by $dH_{ac}(t)/dt$) is proportional to the flux in the sample when the applied ac field is at the peaks, as illustrated by the shaded area in Figure 2.2.3(a) [48]. The out-of-phase signal $V_0\mu_1''$, though, is proportional to the flux trapped in the sample as the ac field crosses zero, as in Figure 2.2.3(b). In fact, the out-of-phase component of the fundamental permeability μ_1'' is related to the ac loss in the sample per unit volume per cycle of ac magnetic field [49], denoted by W_v, by

$$\frac{\mu_1''}{4\pi} = \chi'' = \frac{W_v}{\pi H_1^2} \ . \tag{2.2.25}$$

It will then not be difficult to see that when the ac magnetic field amplitude is larger than the penetration field, $H_1 > H^*$, the rate of increase of W_v as a function of H_1 is not as fast as when $H_1 < H^*$. In fact, according to the critical state model, μ_1'' shows a peak at H^* when calculated as a function of the ac field amplitude H_1, regardless of

the different versions. This peak of μ_1'' is directly measurable in experiments and thus reduces the number of unmeasurable fitting parameters in our generalized critical state equation, Eqn. (2.2.10), from three to two, namely β and H_0.

Finally, we mention that while the generalized critical state model will likely be appropriate for dense sintered samples, thin films and crystals, it is not expected to be valid for the intergranular component for (loosely packed) powdered samples, which cannot support bulk, macroscopic circulating currents.

2.3 Figure Captions and Figures of Chapter 2

Figure 2.1.1. (a) The metallographically observed grains in a powdered copper-oxide superconductor are modelled to behave as (b) an ensemble of superconducting loops with different loop areas and orientations. (c) Metallographically observed grains may be composed of subgrains in electrical contact with each other through weak links. (d) A prototype superconducting loop in the ensemble (b), modeling the grain (c). The loop has area S_0 and junction area s_0.

Figure 2.1.2. (a) Harmonic power $P(n\omega)$ versus h_{dc}, computed from Eqns. (2.1.4) and (2.1.6) with $\sigma = 0$, corresponding to a single loop, with no averaging; figure is valid for all values of h_1 and odd n, and shows periodicity $\triangle h_{dc} = \pi$ due to flux quantization of the loop. (b) Harmonic power $P(n\omega)$ versus h_{dc} computed from Eqns. (2.1.4) and (2.1.6) with $\sigma = 2$, $h_1 = 5$, and $n = 1$; sample averaging washes out the sharp dips of (a). (c) Harmonic power $P(15\omega)$ versus h_{dc} computed from Eqns. (2.1.4) and (2.1.6) for $\sigma = 2$, $h_1 = 5$, and $n = 15$; sample averaging does not wash out the sharp dips. (d) Harmonic power $P(2\omega)$ versus h_{dc}, computed from Eqn. (2.1.12) for $n = 2$, $h_1 = 0.5$, and a monotonically decreasing distribution function $F(A)$. (e) Harmonic power $P(2\omega)$ versus h_{dc}, computed from Eqn. (2.1.4) for $n = 2$, $h_1 = 0.5$, and same $F(A)$ as in (d). (f) Harmonic power $P(15\omega)$ versus h_{dc}, computed for the first-order model, Eqn. (2.1.13), with $h_1 = 5$, $\kappa = 0.3$, and $L_0 = 0.35$, using $F(A)$ from Eqn. (2.1.6) and $\sigma = 2$.

Figure 2.2.1. (a) A plot of local magnetic fields $H(r)$ for a long cylinder of radius R according to the Bean model, Eqns. (2.2.6) and (2.2.12), for applied magnetic field equal to 0, $0.5H^*$, H^*, $1.5H^*$ and $2.0H^*$. (b) A plot of local magnetic fields $H(r)$ according to the Bean model after an external magnetic field of H_{max} has been applied and then removed. The shaded area represents the fact that magnetic flux is trapped after the field is removed.

Figure 2.2.2. (a) A plot of local magnetic fields $H(r)$ for a long cylinder of radius R according to the Kim-Anderson model, Eqns. (2.2.8) and (2.2.13), for applied magnetic field equal to 0, $0.5H^*$, H^*, $1.5H^*$ and $2.0H^*$. (b) A plot of local magnetic fields $H(r)$ according to the Kim-Anderson model after an external magnetic field of H_{max} has been applied and then removed. The shaded area represents the fact that magnetic flux is trapped after the field is removed. (c) Same as (a), except that the generalized critical state model, Eqns. (2.2.10) and (2.2.11), is used, with $\beta = 1.8$, $H_0 = 3.0$. (d) Same as (b), except that the generalized critical state model, Eqn. (2.2.10), is used, with $\beta = 1.8$, $H_0 = 3.0$.

Figure 2.2.3. (a) The instantaneous flux distribution in a cylindrical specimen at the time the ac field $H_{ac}(t) = H_1 \cos(\omega t)$ is at its peak. (b) Instantaneous flux distribution at the time when $H_{ac}(t) = 0$. The signal from the detector is proportional to the shaded area whether the flux profile is linear or not. [48]

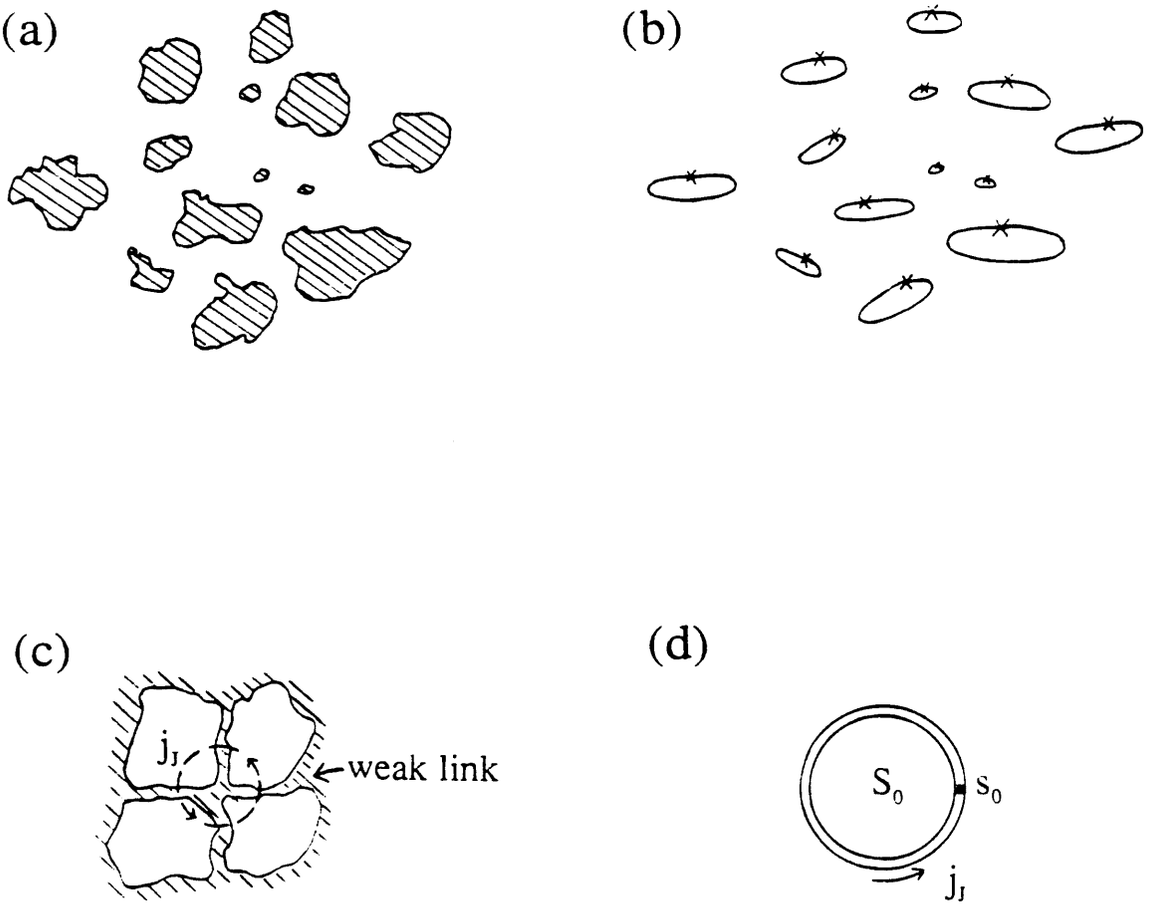


Figure 2.1.1

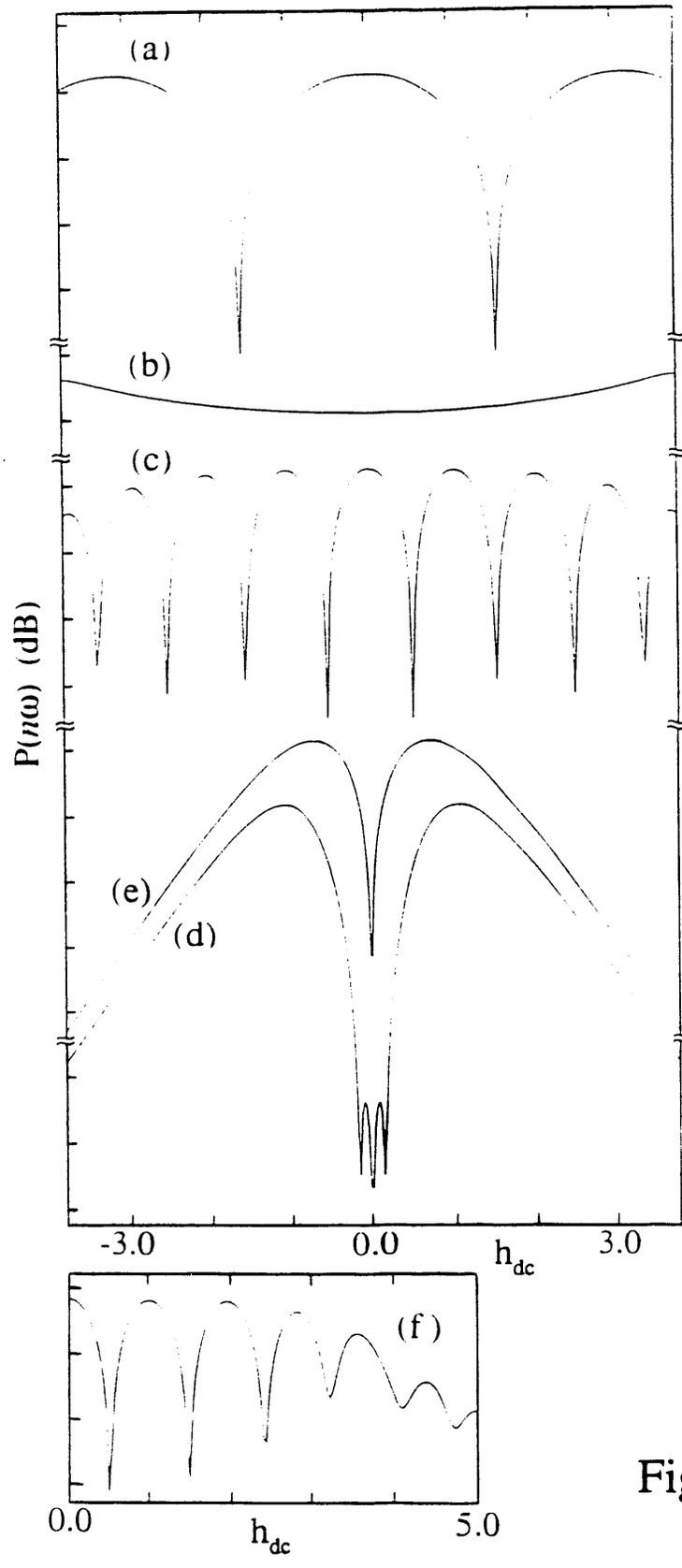


Figure 2.1.2

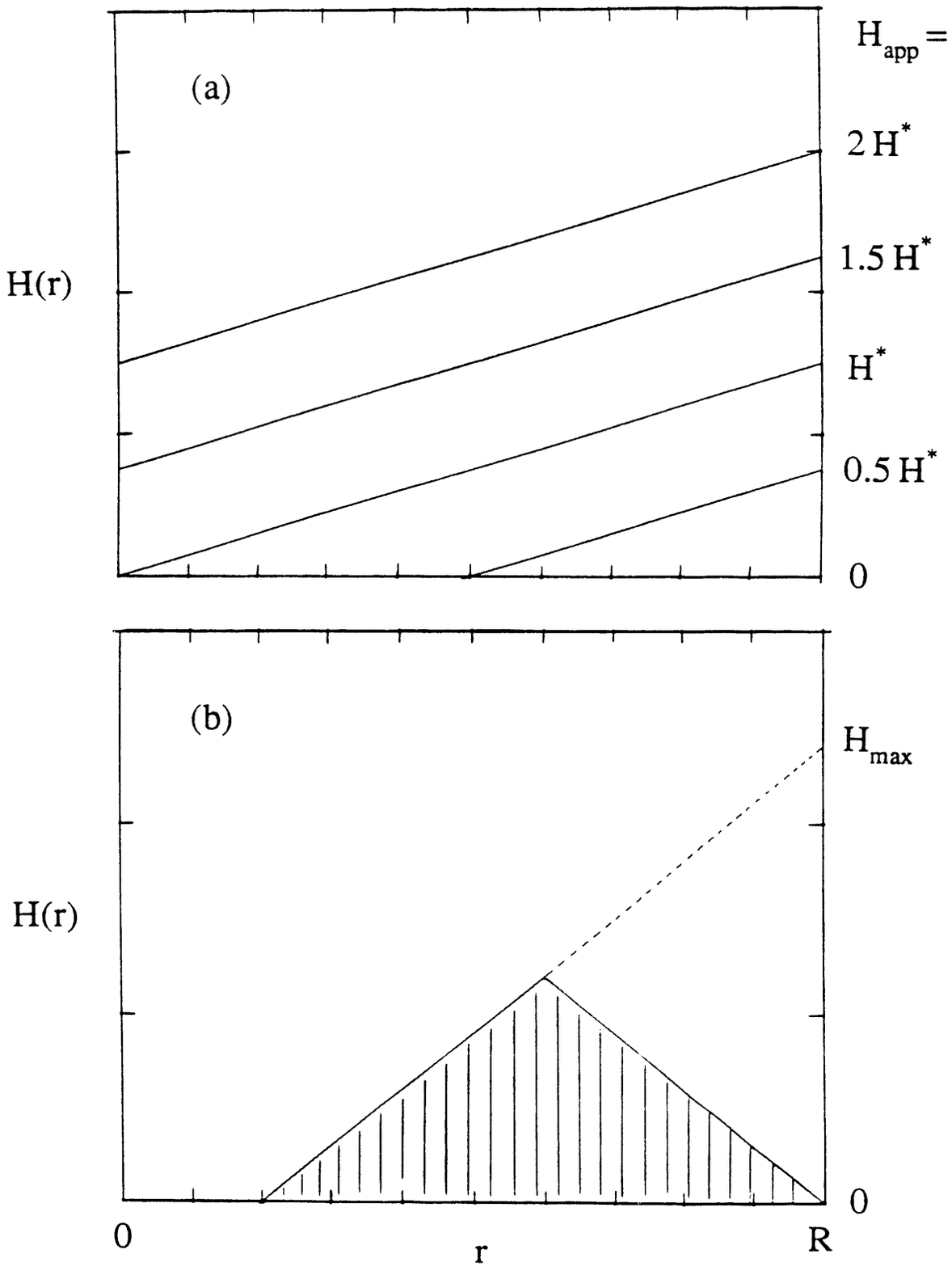


Figure 2.2.1

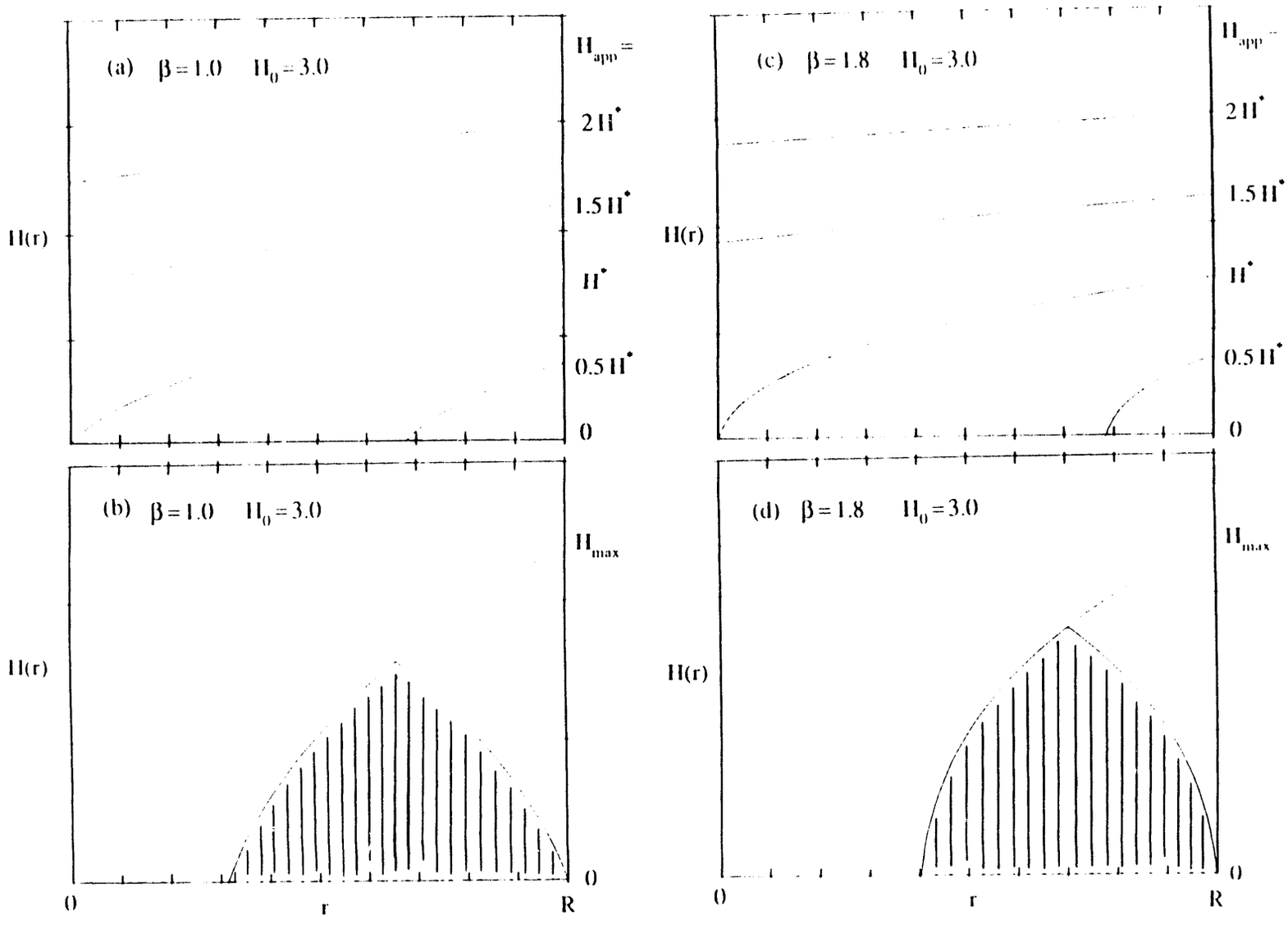

(a) β = 1.0 H0 = 3.0
Happ =
2 H*
1.5 H*
H*
0.5 H*
0
H(r)
(b) β = 1.0 H0 = 3.0
Hmax
H(r)
0
r
R
(c) β = 1.8 H0 = 3.0
Happ
2 H*
1.5 H*
H*
0.5 H*
0
H(r)
(d) β = 1.8 H0 = 3.0
Hmax
H(r)
0
r
R

Figure 2.2.2

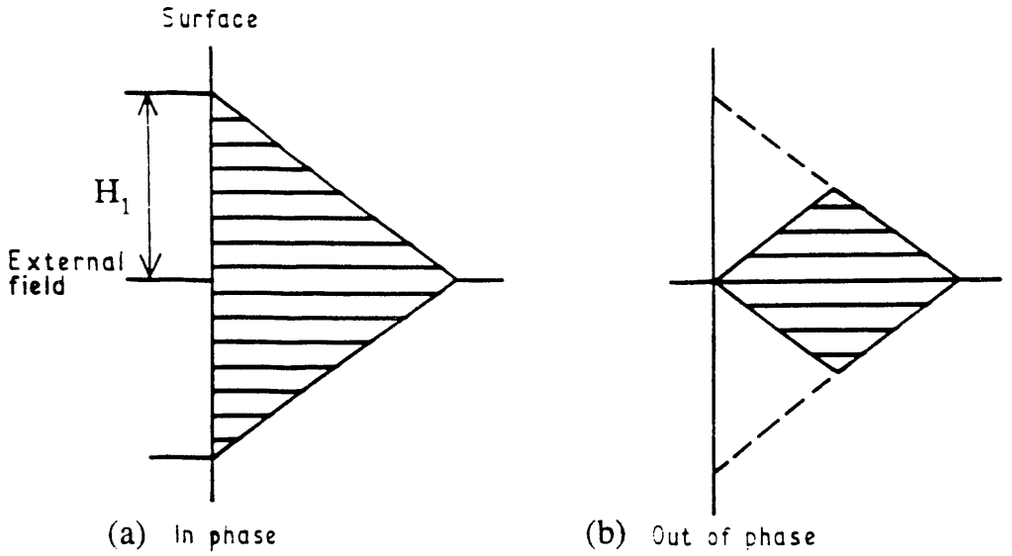


Figure 2.2.3

Chapter 3 Experimental Procedures

In this chapter, details of the experimental procedures will be presented. In Section 3.1, the samples used in the experiments presented in this thesis and relevant publications will be listed and described. In Section 3.2, details about the experimental setup and apparatus will be presented. A table will be given which provides information about the magnetic coils used in the experiments. The data acquisition system will be described in Section 3.3. Actual computer codes written for automated data acqusition will be presented in Appendix C.

3.1 Samples

In Table 3.1.1, the various samples used in the experiments are listed by sample numbers.

Table 3.1.1 Samples used in the experiments.

Sample number	Type of sample	Dimensions	Source
C-15	YBCO powder	Grain radii range from below 1 to 10 microns.	Zettl Group at Berkeley
C-46N	YBCO ceramic cylinder density = 5.46 g/c.c.	3.07 mm diameter x 22.9 mm length	National Superconductor Inc. (Catalog No. B-4015C)
C-48B	1 volume of YBCO powder mixed with 1 volume of 1 micron grit and 1 volume of 0.1 micron grit alumina powder	Grain radii range from 1 to 60 microns, with rough average of about 10 microns.	Same original batch as C-46N
C-50	BSCCO(2212) single crystal	(approx.) 2.8 mm x 3.3 mm x 10 microns thickness	Zettl Group
C-51	pulsed-laser ablated YBCO thin-film deposited on strontium titanate	(approx.) 3 mm x 3 mm x .5 micron thickness	Paul Berdahl Group at Lawrence Berkeley Laboratory

The $YBa_2Cu_3O_{7-\delta}$ ceramics and powdered samples were produced either by Professor Alex Zettl, Dr. Lincoln Bourne, and Mr. C. M. Kim here at Berkeley, or by National Superconductor Inc. The ceramics were made by standard procedures [50] of grinding together stoichiometric mixture of Y_2O_3, $BaCO_3$ and CuO, calcinating in oxygen, sintering, and finally annealing in oxygen. The powdered samples, C-15 and C-48B, were made from the ceramic by grinding the latter in an agate mortar. Ceramic samples in cylindrical forms, such as C-46N, were made by cutting them first into rectangular shape of appropriate dimensions, and then rolling them on fine Al_2O_3 paper into cylindrical shape of the desired radii.

The Y-Ba-Cu-O thin-film C-51 was made by pulsed laser ablation on $SrTiO_3$ by Dr. Paul Berdahl, Dr. Richard Russo and Mr. Ron Reade at Lawrence Berkeley Laboratory. It is about .5 μm thick and its diameter is $\sim$ 3 mm, and is oriented with the c-axis perpendicular to the plane of the film. It has a critical temperature of 87 K.

The Bi-Sr-Ca-Cu-O single crystal C-50 was made by Professor Alex Zettl, Mr. Gabriel Briceño, and Dr. Angelica Behrooz at Berkeley. The crystal-growing procedure is described in detail in Ref. [51]. The crystal size is about 2.8 $\times$ 3.3 mm^2, with a thickness of $\sim$ 10 μm. Its transition temperature, both by four-probe dc resistance and dc magnetic susceptibility measurements, is approximately 88 K; full resistive transition width is of order 2–3 K.

3.2 Experimental Setup

The method we use to study nonlinear electrodynamical behavior consists essentially of subjecting a high-temperature superconducting sample to an ac magnetic field generated by a copper solenoid, and observing and investigating the responses of the sample through the voltage which the sample induces into a surrounding copper "receiver" coil. A dc magnetic field and a slowly scanning magnetic field may be added coaxially onto

the ac magnetic field. The total applied field is thus

$$H_{app}(t) = H_{dc} + H_1 \cos\omega t + H_{scan}(t) . \qquad (3.2.1)$$

All the fields are produced by coaxial copper solenoids immersed in liquid nitrogen and thus kept at 77 K for temperature stability. The nitrogen Dewar is encased in an annealed hypernom magnetic shield, which reduces the residual Earth's field to the milliOersted range.

A diagram of the experimental setup is shown in Figure 3.2.1. Specifications of the magnetic coils used are listed in Tables 3.2.1 and 3.2.2.

Table 3.2.1 Coils used in the experiments (I).

Coil	Purpose	Wire size	No. of layers
1	AC magnetic field ("Transmittor coil")	No. 32	4
2	Slow scanning magnetic field	No. 36	32
3	DC bias magnetic field	No. 32	6
4	Additional DC magnetic field	No. 36	4
5	Signal pick-up coil (Top "receiver coil")	No. 36	4
6	Balancing coil (Bottom "receiver coil")	No. 36	4+

The ac magnetic field [44], $H_{ac}(t) = H_1 \cos(\omega t)$, has a frequency range of $f = \omega/2\pi \approx 10^2$ to 10^5 Hz. The field is produced by a copper solenoid 10.2 cm long and 1.74 cm diameter; its number of turns is 1667, divided into four layers and its inductance is about 8.2 mH (Coil #1 in Tables 3.2.1 and 3.2.2). The drive-coil is driven by a very stable, synthesized function generator (HP model 3325A). In case a high ac field ($30 \leq H_1 \leq 600$ Oe) is required, the function generator will drive the solenoid through a very linear NAD ac power amplifier (model number 2100), which has residual harmonic power at about 80 dB below the fundamental mode. In order to prevent overheating or even burning the ac-drive-coil when high ac field is being

Table 3.2.2 Coils used in the experiments (II).

Coil	Length	Diameter (inner)	Coil constant	Resistance at room temp	No. of turns
1	10.2 cm (4.00 inches)	1.75 cm (0.687 inches)	186.3 Oe/A (at 85 Hz)	52 ohms	1667
2	11.0 cm (4.35 inches)	3.56 cm (1.40 inches)	2508.6 Oe/A	3.89 kilo-ohms	22661
3	11.0 cm (4.35 inches)	3.31 cm (1.30 inches)	328.5 Oe/A	166 ohms	2938
4	11.0 cm (4.35 inches)	3.18 cm (1.25 inches)	318.9 Oe/A	404 ohms	2892
5	1.25 cm (0.5 inch)	1.1 cm (~ 0.45 inch)	N/A	17.5 ohms	324
6	1.25 cm (0.5 inch)	1.1 cm (~ 0.45 inch)	N/A	17.9 ohms	341

generated, special attention has been paid to ensure proper circulation of liquid nitrogen around the coil, since the cryogen is constantly being boiled into gaseous form by the coil when it is generating high ac fields.

For measurements of harmonic power versus superposing dc magnetic field, the "dc" field is scanned at a cycle time larger than or equal to 100 seconds, using Coil #2 in Tables 3.2.1 and 3.2.2 . This scanning field $H_{scan}(t)$ is produced by another HP3325A synthesizer, which can produce a sawtooth waveform. The maximum span of the scanning dc field is about ± 100 Oe. If more dc field is required, it can be provided at a fixed value by solenoids Coil #3 and #4.

The signal voltage induced by the superconducting samples into the receiver coil is of the form $V(t) = \sum V_n(t)$, where $V_n(t) = A_n \sin(n\omega t) + B_n \cos(n\omega t)$, $n = 1, 2, \ldots$. The signal voltage can be processed by an analog spectrum analyzer with a 100-dB dynamic range (HP model 3585A) , to yield the power spectral components $P(nf) \propto (A_n^2 + B_n^2)$. The signal can also be processed by a lock-in amplifier (PARC

model 5209), which can distinguish the individual phase components, A_n and B_n, at $n = 1$ and $n = 2$.

Two arrangements of receiver coil are used in our experiments. In the *two-coil method*, the superconducting sample is located in a solenoid of length 1.25 cm and diameter 1.15 cm (coil no. 5). Voltage signal from the sample picked up by this solenoid is subtracted externally by that of a similar but empty solenoid, coil no. 6, which is being driven by the same ac field. Because of the slight differences in dimensions of Coil #5 and #6 due to machining precision, the number of turns in the two coils are intentionally set to different values. When both coils are empty, and connected in opposition, their total signal voltage is balanced out to within 0.5% of their individual values without additional aid of circuit balancing. In case a better balance is desired, a simple balancing circuit, shown in Figure 3.2.2, is used. This circuit can balance both the in-phase and in-quadrature components to within 100 ppm of the individual coils' pick-up voltage values. The resulting voltage signal from the "two-coil" receiver is equal to the time-derivative of the sample magnetization dM/dt. The fundamental mode in-phase and in-quadrature components of the lock-in output voltage, when normalized by the ac field amplitude H_1, are proportional to χ_1' and χ_1'', respectively, where $\tilde{\chi_1} \equiv \chi_1' - i\chi_1''$, is the fundamental mode complex susceptibility of the sample.

In the *one-coil method*, a cylindrical bar of ceramic superconducting sample is closely wound directly with a single receiver coil of no. 40 copper wire. The receiver (cross-sectional area A, number of turns $= N = 78$) generates a signal voltage $V(t)$ proportional to the time-derivative of the instantaneous induction field $\langle B(t) \rangle$ averaged over the whole sample. The signal voltage can be expanded in a Fourier series

$$V(t) = \left[\frac{NAH_1\omega}{c}\right] \mu_{eff} \sum_{n=1}^{\infty} \left[n\mu_n' \sin(n\omega t) - n\mu_n'' \cos(n\omega t) \right] , \qquad (3.2.2)$$

where the bracketed term outside the summation is the signal amplitude when the sample is normal. For ceramic samples, μ_{eff} is the effective permeability of the polycrystalline

sample, given by (Eqn. (2.2.3))

$$\mu_{eff}(T) = f_n + f_s \left[1 - F\left(\frac{R_g}{\lambda_g(T)} \right) \right] , \qquad (3.2.3)$$

where f_n and f_s are the intergranular (including voids) and intragranular volume fractions and $f_n + f_s = 1$. R_g and λ_g are the grains' (averaged) radius and London penetration depth respectively, and $F(R_g/\lambda_g)$ is the factor by which the magnetic flux penetration suppresses a grain's magnetization below that expected for complete Meissner-state flux exclusion. More discussion about this will be given later in Chapter 4. The Fourier components are the real and imaginary parts of a complex ac permeability $\tilde{\mu}_n = \mu_n' - i\mu_n''$ for the n^{th} harmonic; they are related to the ac susceptibility $\tilde{\chi}_n = \chi_n' - i\chi_n''$ by $\tilde{\mu}_1 = 1 + 4\pi\tilde{\chi}_1$, and $\tilde{\mu}_n = 4\pi\tilde{\chi}_n$ for all $n > 1$.

Most data presented in this thesis are taken on samples immersed in liquid nitrogen and kept at $T = 77$ K. The only exception are the data for the ceramic $YBa_2Cu_3O_7$ cylinder C-46N, in which clear signals of inter- and intragranular supercurrents are detected. For this sample, the temperature is varied from 77 K to above $T_c \approx 92$ K. To do this a simple temperature-control system is made, as follows.

Temperature control system. A schematic diagram of the temperature control system is shown in Figure 3.2.3. The sample C-46N is placed in a 5 mm o.d., 4 mm i.d. quartz sample tube, free of magnetic impurities, produced by Wilmad Co for use in an EPR spectrometer. Warm nitrogen gas ($T \approx 160$ K) is generated from a separate liquid nitrogen storage dewar with a controlled resistive heater and forced through a teflon tube into the sample tube. Because the diameter of the sample is about 1 mm smaller than the i.d. of the sample tube, the whole sample will be in thermal contact with the warm nitrogen gas. The 5 mm o.d. sample tube is then sealed and inserted into a 9 mm od. and 7 mm id. quartz tube. The space between the two quartz tubes is packed with thermally insulating materials. Such a double-tube with insulating packing prevents direct heat-sinking from taking place when the whole double-tube is immersed in liquid

nitrogen. However, the double-sample-tube-complex is intentionally designed to have a less-than-perfect thermal insulation to allow the originally 160 K, warm nitrogen gas to be cooled by the outside liquid nitrogen to the desired ambient temperature for the sample within a convenient time-scale. The final temperature of the sample will be determined by the flow-rate of the gas — the higher the flow-rate, the warmer the temperature of the sample. The coarse control of the flow-rate of the gas is managed by means of a needle-valve at the outlet of the gas. The fine control is provided by the heater in the liquid nitrogen storage dewar which supplies the warm nitrogen gas. The heater boils the liquid nitrogen and builds a pressure in the storage dewar; by controlling the power of the heater, one controls the pressure inside the storage dewar and thus the flow-rate of the warm gas. Such a simple system manages to control the temperature of the sample from 77 K to well above $T_c \approx 92$ K, and is stable to within 0.2 K. Temperature is monitored by a copper-constantan (3 mil diameter) thermocouple with the tip attached by stycast to the bottom of the sample. It is so located to avoid direct blowing by the warm gas onto the couple junction, which may cause a higher-than-actual temperature reading. The reference junction of the thermocouple is liquid nitrogen and the voltage is read by a digital multimeter (Keithley 197) which is monitored through GPIB (General Purpose Interface Bus) by the computer.

Due to the size of the sample, though, it takes about 20 minutes for the sample and the thermocouple to come to equilibrium every time the temperature is changed. Also note that while the temperature of the sample is varied, all the magnetic field coils in the system are stably maintained at 77 K. At a set temperature, desired data are taken rapidly by the following GPIB computer system.

3.3 Data Acquisition

The data presented in this thesis are either taken in the analog mode by an x-y recorder, or in the digital mode by an AT-compatible computer made by Fountain

Technologies Inc.

A block diagram for the analog mode is shown in Figure 3.3.1. This mode is used mainly for measuring the harmonic power $P(nf)$ generated by the superconducting samples as a function of the "dc" field superposed on the ac magnetic field. The "dc" field is actually $H_{dc} + H_{scan}(t)$, a slowly scanning field at a frequency $f_{scan} \leq 0.01$ Hz, plus a small true dc field preset to balance out any residual Earth's magnetic field. The scanning is provided by a synthesizer with a sawtooth output waveform. The output voltage from the scanning synthesizer is used to drive both the scanning dc-coil and the x-axis of the x-y recorder.

The y-axis of the x-y recorder is driven by the "video-output" of the spectrum analyzer. This output produces a voltage signal proportional to the power of the harmonic chosen to be measured. So as the dc field is varied by the scanning synthesizer, $P(nf)$ as a function of H_{dc} can be plotted.

An analog spectrum, $P(nf)$ versus frequency, of the sample's signal can also be easily taken by the x-y recorder. In this case, with the dc field being fixed at a desired value, the x- and y-input of the x-y recorder are driven respectively by the "x-output" and "y-output" of the spectrum analyzer. The recorder will then generate a hard-copy of the spectrum currently on the CRT of the spectrum analyzer.

A block diagram for the digital mode is shown in Figure 3.3.2. This mode is used mainly for measuring either the harmonic power $P(nf)$ using the spectrum analyzer or the two components of the complex permeability using the lock-in amplifier as a function of the ac magnetic field amplitude H_1. In this mode, the computer automates the data acquisition by means of a National Instrument GPIB-PCII card. Among the electronic apparatus, the Keithley 197 DMM's, the lock-in amplifier, the synthesizers and the spectrum analyzer all have built-in GPIB capability. This system reads and digitizes these parameters: H_{dc}, H_1, V_n', V_n'', $P(nf)$, and the thermocouple voltage V_T.

These can be read essentially simultaneously into data files. Control programs and file structures are described in Appendix C.

When the harmonic power $P(nf)$ is to be measured as a function of H_1, the computer program can systematically step up the voltage output of the synthesizer which is driving the ac-coil, probably through the NAD ac amplifier, and thus step up H_1. The current through the precalibrated ac-coil is monitored by the voltage across a monitoring resistor which is immersed in liquid nitrogen for stability. This ac-field-monitoring voltage is measured by a Keithley DMM and read by the computer through GPIB. The power of the chosen harmonics $P(nf)$ is also read by the program from the spectrum analyzer through its GPIB. Caution should be exercised to avoid driving the NAD amplifier to nonlinearity. The program checks and decides if the output of the driving synthesizer is exceeding the maximum tolerance of the NAD amplifier; if so, it will pause and let the experimenter manually step down the synthesizer output and, to compensate, step up the amplification factor of the NAD.

When the complex permeability components μ_1' and μ_1'' are to be measured as a function of H_1, a little more caution is needed. Correct measurements of the components, especially μ_1'', are sensitive to the correct phase-setting of the lock-in amplifier. In this thesis, the phase of the measured μ_1' is first determined, in the case of 2-coil receiver, by setting the lock-in to be in phase with the induced voltage ($\propto H_1 \omega \sin \omega t$) of the empty and unbalanced upper receiver coil. In the case of the 1-coil receiver wound directly on the cylindrical ceramic superconducting sample, the sample is brought to well above its critical temperature and the resulting phase at maximum voltage is defined to be the phase of μ_1'.

There is actually another complication about correct phase-setting which is rather unexpected. As it turns out, if one is using a fixed output of the HP3325A synthesizer, the relative phase between the "SIGNAL" output and the "SYNC OUT" of the synthesizer is fixed to a value close to but not equal to 0°. However, this relative phase changes

when the "SIGNAL" value is changed, up to as much as ±5°. In other words, in a phase-sensitive measurement of μ_1' and μ_1'' *as a function of* H_1, the "SYNC OUT" of the ac-drive synthesizer cannot be used as the reference to the lock-in, because the phases of the measured μ_1' and μ_1'' would be constantly changed as H_1 is being stepped up. This behavior of the synthesizer causes some inconvenience in automating the data-taking process because, to make sure that the phases of the two measured components are still correct, one would have to check the phases after every data point of a different H_1 value. This of course could be done by using a Lissajous figure on an oscilloscope.

It is fortunate that the changes in the phase of the synthesizer with changes in its output are quite systematic within a particular decade of output voltage. I found that this relative phase between the "SIGNAL" output and the "SYNC OUT" has about the same value when the "SIGNAL" is at, say, 0.02 V_{pp}, 0.20 V_{pp} and 20 V_{pp}. Also, the unit to unit difference in this relative phase change does not vary much, at least at low frequencies ($\sim 100 - 500$ Hz). So, to automate the system and to eliminate constant checking of the phases, a second synthesizer set to the same frequency is phase-locked to the ac-drive synthesizer and its output, which is set to vary according to that of the ac-drive synthesizer, is used as the lock-in reference. The output of this second synthesizer is set to vary within the input-voltage tolerance of the lock-in reference. For instance, when the ac-drive synthesizer's output is 20 V_{pp}, the phase-locked reference synthesizer output will be set to 2.0 V_{pp}. This compensation method enables the full-automation of the data-taking process of the μ_1' and μ_1'' measurements as a function of H_1.

3.4 Figure Captions and Figures of Chapter 3

Figure 3.2.1. Diagram of the experimental mutual inductance bridge apparatus. Details of the magnetic coils are given in Tables 3.2.1 and 3.2.2.

Figure 3.2.2. Balancing circuit for the receiver coils (Coils #5 and #6 in Figure 3.2.1). The circuit can balance the in-phase and in-quadrature signals to within 100 ppm of the individual coils' pick-up voltage values.

Figure 3.2.3. Temperature control system (for ceramic Y-Ba-Cu-O sample, no. C-46N). The temperature is controlled by the rate of flow of the warm nitrogen gas; the temperature increases as the rate of flow is increased. The rate is coarsely controlled by the needle valve and finely controlled by the power resistor in the storage dewar.

Figure 3.3.1. Block diagram for the analog data acquisition system. The coils and the sample are immersed in liquid nitrogen dewar, represented by the dashed lines.

Figure 3.3.2. Block diagram for the digital data acquisition system: GPIB (General Purpose Interface Bus). The coils and the sample are immersed in liquid nitrogen dewar, represented by the dashed lines. The copper-constantan thermocouple, however, is not directly immersed in liquid nitrogen; details about temperature control are shown in Figure 3.2.3. The GPIB control programs are described in Appendix C.

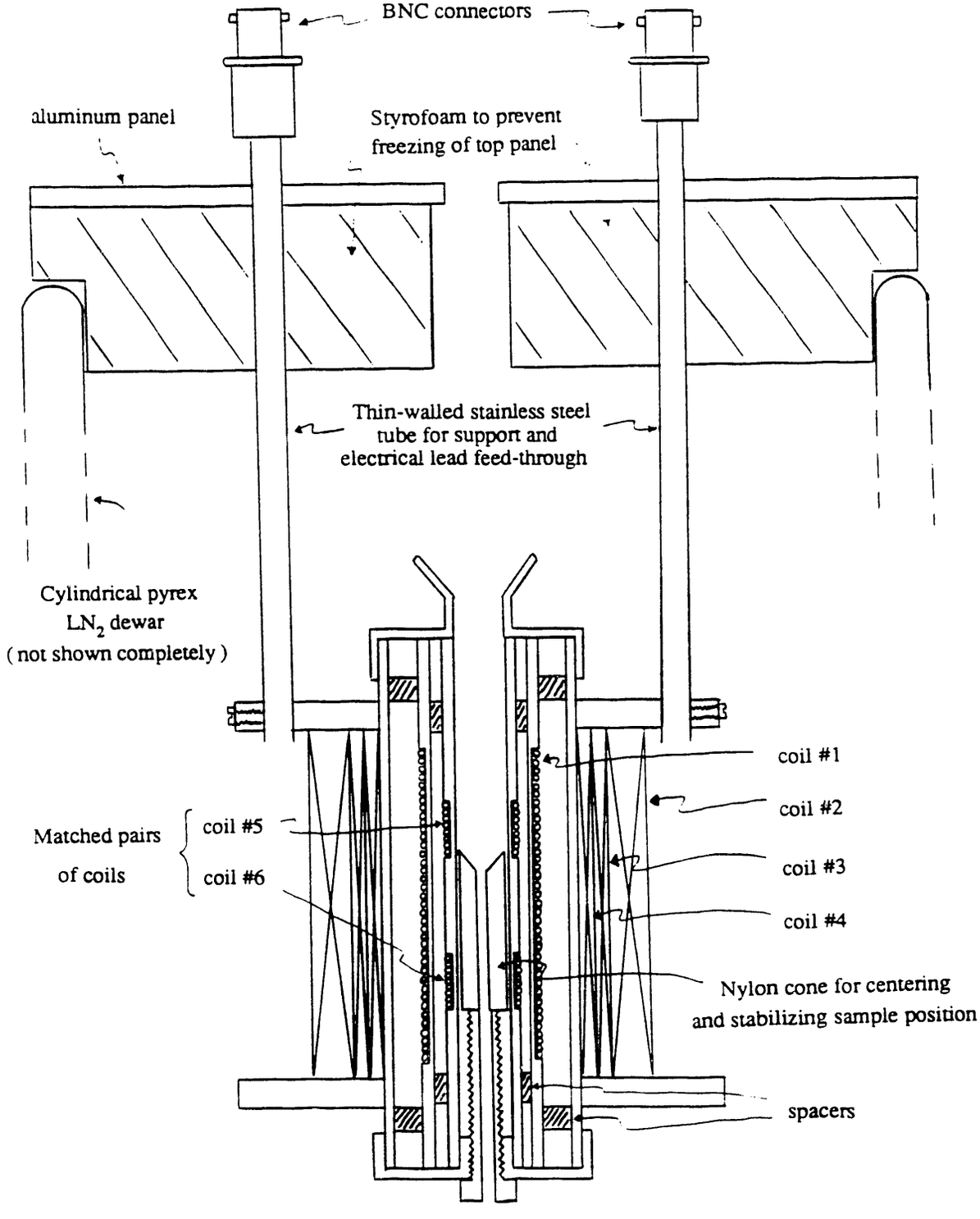


Figure 3.2.1 (Not to scale)

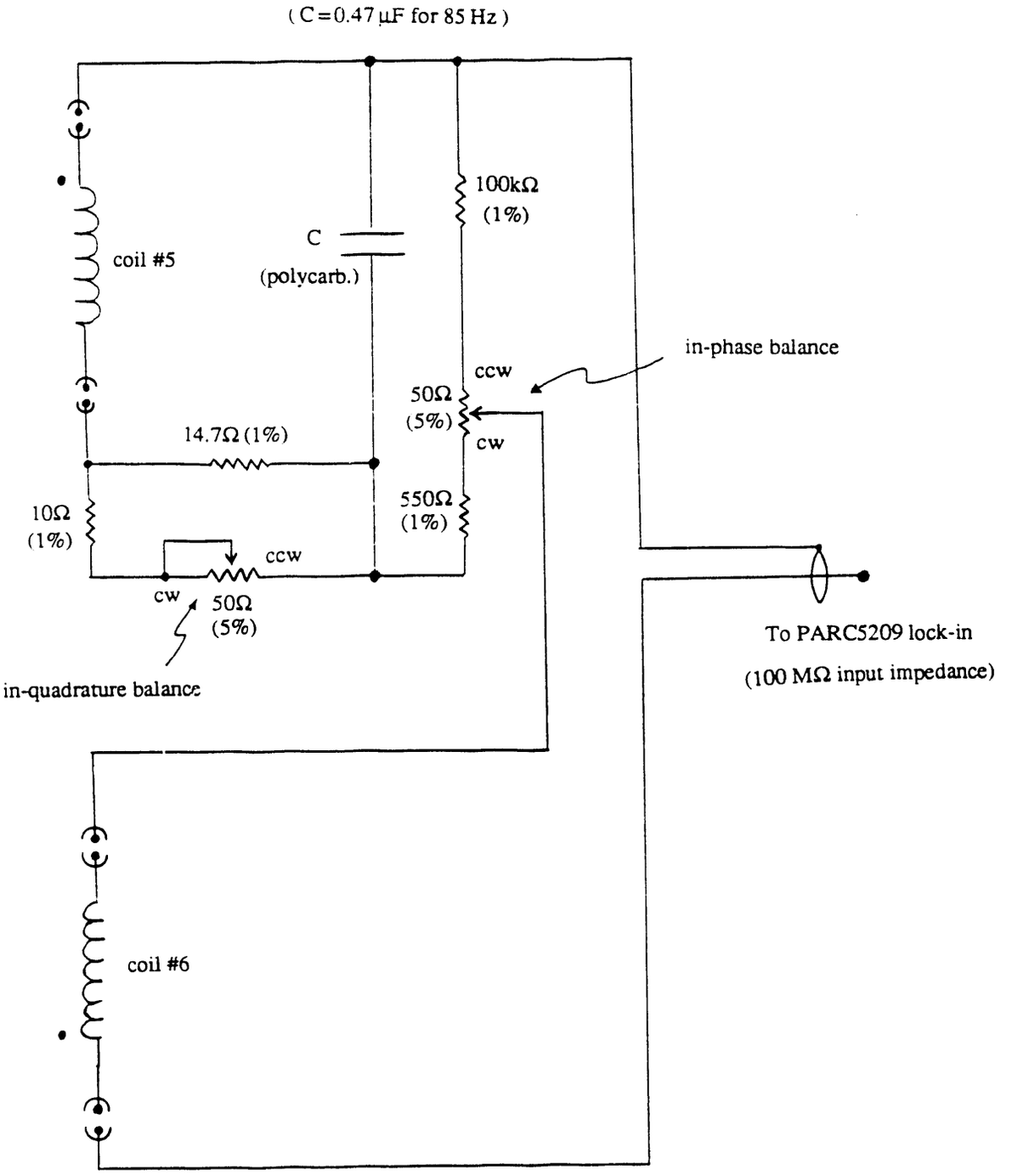

(C = 0.47 μF for 85 Hz)
100kΩ
(1%)
C
(polycarb.)
coil #5
in-phase balance
ccw
50Ω
(5%)
cw
14.7Ω (1%)
550Ω
(1%)
10Ω
(1%)
ccw
cw
50Ω
(5%)
in-quadrature balance
To PARC5209 lock-in
(100 MΩ input impedance)
coil #6

Figure 3.2.2

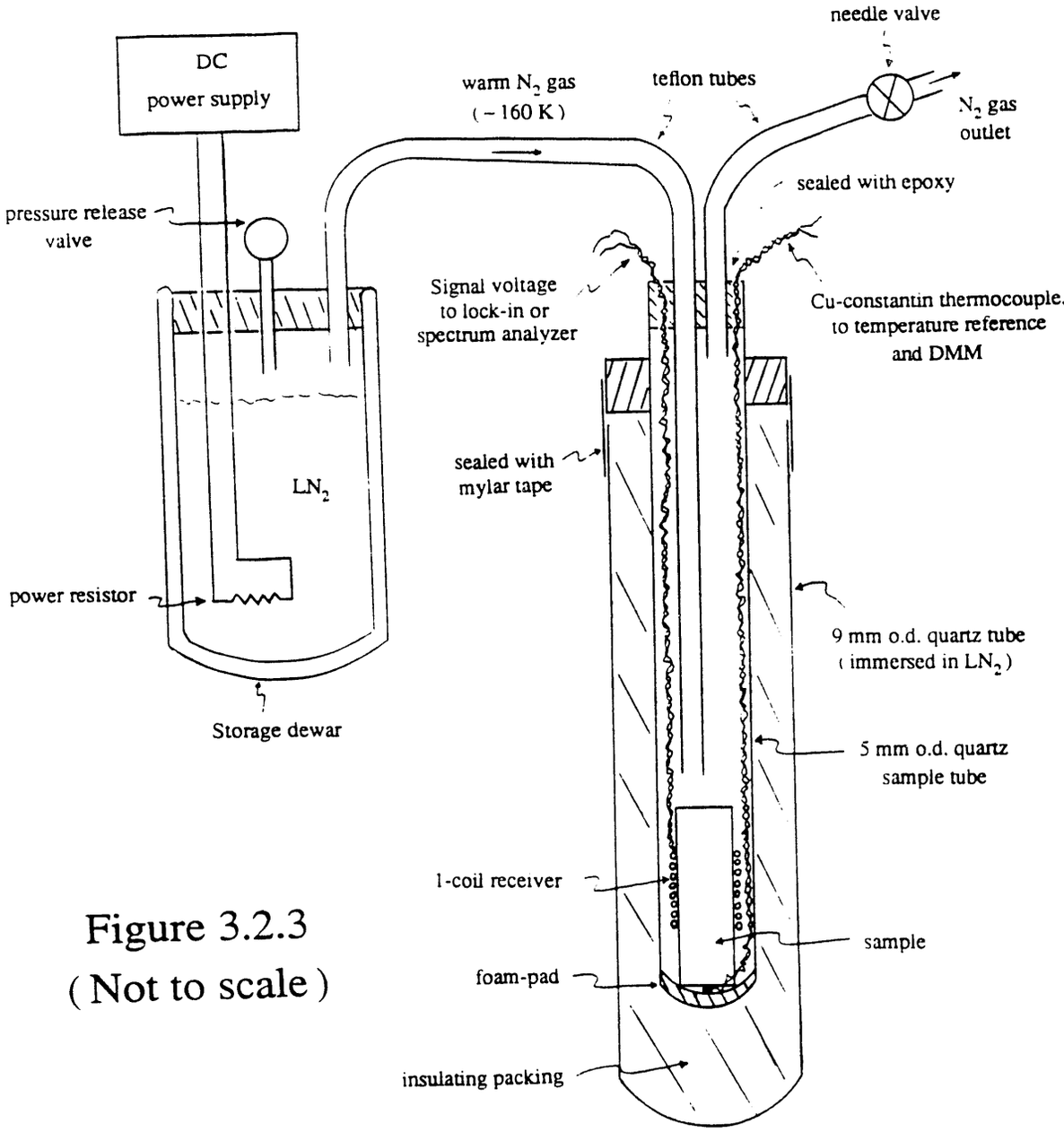


Figure 3.2.3
(Not to scale)

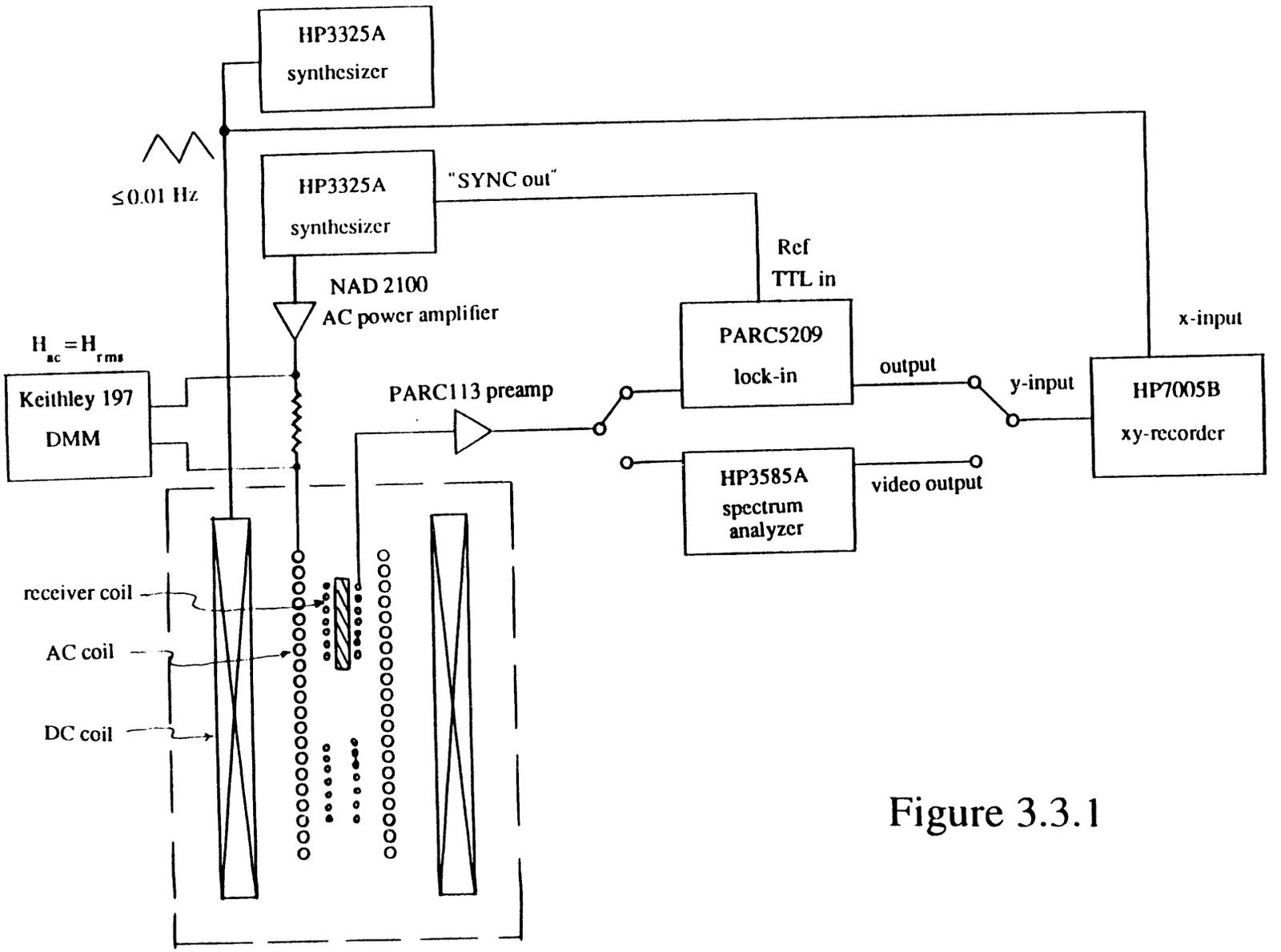

HP3325A
synthesizer
≤0.01 Hz
HP3325A
synthesizer
"SYNC out"
Ref
TTL in
NAD 2100
AC power amplifier
x-input
$H_{ac} = H_{rms}$
Keithley 197
DMM
PARC113 preamp
PARC5209
lock-in
output
y-input
HP7005B
xy-recorder
HP3585A
spectrum
analyzer
video output
receiver coil
AC coil
DC coil

Figure 3.3.1

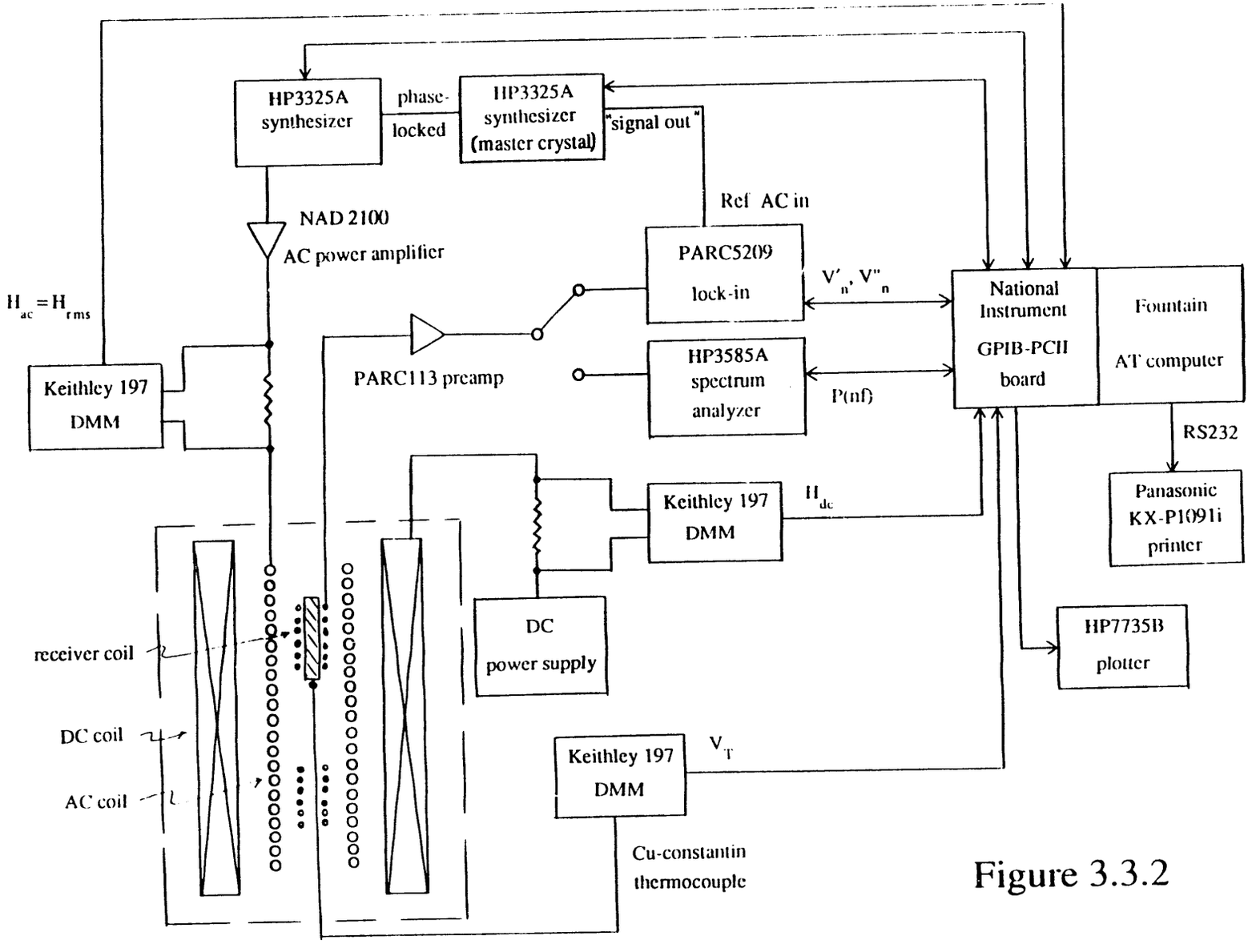

HP3325A synthesizer
phase-locked
HP3325A synthesizer (master crystal)
"signal out"
Ref AC in
NAD 2100
AC power amplifier
PARC5209 lock-in
V'n, V"n
National Instrument GPIB-PCII board
Fountain AT computer
Hac = Hrms
Keithley 197 DMM
PARC113 preamp
HP3585A spectrum analyzer
P(nf)
RS232
Panasonic KX-P1091i printer
Keithley 197 DMM
Hdc
DC power supply
HP7735B plotter
receiver coil
DC coil
AC coil
Keithley 197 DMM
VT
Cu-constantin thermocouple

Figure 3.3.2

Chapter 4 Results and Analysis

In this chapter we present the results of a series of experimental investigation on the nonlinear electrodynamics of high-temperature superconductors in both powdered and bulk form. The samples and their properties are summarized in Table 3.1.1. In these experiments, investigation of high harmonic power generation by the samples plays an important role. These high harmonics are generated by the nonlinearity in electrodynamical behavior of the samples. Other than being an interesting subject of their own, the high harmonics provide a severe test of the models that we are going to use to explain the experimental results. As described in Chapter 3, the signal of the experiments come from a receiver coil in which a superconducting sample is located. The signal is of a form $V(t) = \sum_n V_n(t) = \sum_n A_n \sin(n\omega t) + B_n \cos(n\omega t)$ and is processed by either an analog spectrum analyzer or, for phase-sensitive detection, a lock-in amplifier.

In Section 4.1, all of the data presented are taken on a powdered Y-Ba-Cu-O sample (C-15) at $T = 77$ K. The data are best explained by the zero-order model as described in Section 2.1.1, owing to the fact that bulk critical currents cannot flow in the powders.

In Section 4.2, the data are mainly taken on a Y-Ba-Cu-O bulk ceramic cylinder. Here, both harmonic power data and the fundamental mode ($n = 1$) complex permeability will emphasized. In these data, one can see unambiguous evidence of the *coexistence* of the inter- and intragranular supercurrent components in a ceramic cylindrical sample of Y-Ba-Cu-O. Estimates of the respective supercurrent critical densities can also be made on these data. The data are explained by the generalized critical state model as described in Section 2.2.

Data taken on the ceramic Y-Ba-Cu-O cylindrical sample at various temperatures will also be presented in Section 4.2. From these data at different temperatures, the phase-locking temperature of the bulk ceramic sample, as separate from the intrinsic

critical temperature of the material, is measured. This measurement gives support to the "gauge-glass" model of describing the superconducting ceramic as a 3-dimensional superconducting array as described in Shih, Ebner and Stroud [52].

Measurements on a pulsed-laser ablated Y-Ba-Cu-O thin film, and on a Bi-Sr-Ca-Cu-O single crystal will be presented in Sections 4.3 and 4.4, respectively. All the measurements are done at 77 K. These data will be compared to the generalized critical state model calculations. In contrast to the Y-Ba-Cu-O ceramic in Section 4.2, both the Y-Ba-Cu-O thin film and the Bi-Sr-Ca-Cu-O single crystal show only one supercurrent component. Estimates of the critical current densities will be presented and suggestions for further work will be made.

4.1 Measurements and models on Powdered Y-Ba-Cu-O

Extensive harmonic generation. When a powdered sample of Y-Ba-Cu-O is cooled in liquid nitrogen in "zero" magnetic field ($H_{dc} \leq 1$ mOe) and driven by an ac magnetic field, signals with harmonic components of the driving frequency will be generated and can be picked up by a receiver coil surrounding the sample. For instance, Figure 4.1.1 shows the harmonic power spectra $P(nf)$ versus the harmonic number n of Y-Ba-Cu-O powder sample C-15 at $T = 77$ K, taken by the two-coil method, with $H_1 = 2.3$ Oe, and $H_{dc} = 0$ and 1 Oe, respectively. Note the symmetry of the even harmonics — at $H_{dc} = 0$, only odd harmonics are generated by the sample; when $H_{dc} \neq 0$, symmetry is broken and even harmonics also appear. We shall discuss more in detail about this symmetry later. Figure 4.1.2 shows the harmonic power $P(nf)$ versus the harmonic number n for sample C-15, also at $T = 77$ K, with $H_1 = 23$ Oe, and $H_{dc} \approx 1$ mOe. The power falls off slowly with n; all odd harmonics up to at least $n = 41$ are clearly observed, superposed on a broad receiver coil resonance at 363 kHz.

Figure 4.1.3 shows the $P(nf)$ versus n data of Figure 4.1.2, corrected for the receiver coil resonance; the slope for large n is 1.9 dB/harmonic. The broken line is that

computed for the zero-order model, Eqn. (2.1.4) with the area distribution

$$F(A) = \frac{\sinh(A\pi/2)}{A\left[\cosh(A\pi) - 1\right]}, \tag{4.1.1}$$

which was previously discussed in Section 2.1.1; the slope for large n is 2.4 kB/harmonic; this model thus gives a reasonable explanation of the slow falloff of the harmonic power. The dotted line, computed for the first-order model, Eqn. (2.1.13), does not fit the data owing to a resonance near $n = 17$ due to the choice of the parameters L_0 and κ in Eq. (2.1.13).

Plots of $P(nf)$ versus H_1 for $n = 3$, 5, and 7 show a roughly cubic dependence on H_1 in the intermediate-H_1 region, for which we have no numerical model, and more complex behavior in the high-H_1 region. We also note that Xia and Stroud [6] have developed a model of superconducting clusters by which they explain the power dependence on n.

Symmetry of harmonic power. The symmetry of the even and odd harmonics with respect to H_{dc} that we mentioned above when describing Figure 4.1.1 can be best illustrated by plotting the harmonic power $P(nf)$ versus H_{dc}. Figure 4.1.4(a) shows the second harmonic power $P(2f)$ generated by sample C-15 as a function of H_{dc} obtained by slowly scanning from $H_{dc} = +20$ to -20 Oe. As pointed out earlier, the second harmonic power becomes essentially nonexistent at $H_{dc} = 0$. The dip in $P(2f)$ at $H_{dc} = 0$ is so sharp that, when shown in expanded scale in Figure 4.1.4(b), one finds that it changes by about 30 dB with a superposition of only 1 mOe, and increases by 85 dB for a dc field change of ~ 1 Oe. This is qualitatively in agreement with the prediction of the zero-order model, based on the symmetry of Eqn. (2.1.4), that $P(nf) \rightarrow 0$ as $H_{dc} \rightarrow 0$ for even n. To observe this very narrow dip it is necessary to both use $H_1 \geq 2$ Oe and to cool in zero field to reduce the remanent local fields due to pinned fluxons. Moreover, there is some experimental evidence that this very narrow dip is an unstable state. If the dc field is scanned up to $H_{dc} > 5$ Oe and back,

the narrow dip cannot be recovered without zero field cooling, possibly the result of pinned fluxons. The system appears to be somewhat unstable, probably at localized sites, against self-symmetry-breaking.

Periodic dips in $P_n(H_{dc})$. Yet another aspect of the nonlinear electrodynamics observable in $YBa_2Cu_3O_7$ powder is shown in Figure 4.1.5, the relative harmonic power for selected harmonics, versus H_{dc}, scanned at a uniform rate for field increasing and then decreasing; the ac field has the relatively large value $H_1 = 23$ Oe. For $n = 2$ the trace is similar to Figure 4.1.4 except for a broader dip at zero field and a larger hysteresis. For higher even harmonics, the same symmetry with respect to $H_{dc} = 0$ as $P(2f)$ holds; the harmonic power goes to a minimum at $H_{dc} = 0$. For odd harmonics, the power has a peak at $H_{dc} = 0$. However, for the higher harmonics, both even and odd, a series of sharp dips in $P_n(H_{dc})$ is observed, approximately equally spaced, with the average spacing $\triangle H_{dc}$ inversely proportional to n. The spacing is given empirically by $\triangle H_{dc} \approx 3H_1/n$, which also is qualitatively predicted by the model as discussed below. These dips are interpreted as evidence for a pseudo flux quantization of superconducting loops in the powdered sample and are a confirmation of the predictions of the model in Section 2.1. For example, Figure 4.1.6 shows $P(nf)$ versus h_{dc}, computed for the zero-order model, Eqns. (2.1.4) and (2.1.9), for the same harmonic numbers as Figure 4.1.5. Since h_{dc} is just proportional to H_{dc} a strong correspondence between experiment and model is readily apparent for all harmonics. The small hysteresis in the data is believed to have an origin in pinning and depinning of fluxons, as discussed by Blazey *et al* [53], and many others.

In the computation for Figure 4.1.6 we used the loop area distribution function Eqn. (2.1.9). As discussed in Section 2.1.1, this monotonically decreasing expression was not chosen arbitrarily, but rather empirically, in order to yield from the integral in Eqn. (2.1.4), for $n = 0$, a dc magnetization of the form $M(H_{dc}) \sim \tanh(H_{dc}/H')$, which is an *approximate* representation of the low-field data reported for granular $YBa_2Cu_3O_7$

at 77 K for which $H' \sim 10$ Oe. Equation (2.1.4) was numerically integrated, with Eqn. (2.1.9), and with limits of integration chosen as $\delta = 0.005$ and $A_{max} = 30.0$. Even though Eqn. (2.1.9) is singular at $A = 0$, it was found that these limits yielded a magnetization within a few percent of the hyperbolic tangent function. Furthermore, we only use Eqns. (2.1.4) and (2.1.9) to compare the *relative* power for various harmonics, or, for a given n, variation of the harmonic signal with h_{dc}; these results are not sensitive to the limits of integration as long as the singularity of $F(A)$ at $A = 0$ is avoided. Equation (2.1.4) with Eq. (2.1.9) also qualitatively predicts the observed shapes of the lock-in voltage signals V_c versus H_{dc}, for $n = 1$ and 2, and the falloff of $P(2f)$ in Figure 4.1.4(a), as H_{dc} moves away from zero.

We also used Eqn. (2.1.4) to compute $P(nf)$ vs h_{dc} for a Gaussian distribution function

$$F(A) = \exp\left\{-\frac{(A-1)^2}{2\sigma^2}\right\}, \tag{4.1.2}$$

finding predictions similar to Eqn. (2.1.9) for large n. However, for small n, predictions do not agree well with experiment. From data like that of Figure 4.1.5, we plot in Figure 4.1.7(a) the average spacing $\overline{\triangle H_{dc}}$ between dips versus harmonic numbers $n = 3$, 4, ..., 30. Except for small n the data are well fit by the expression $\overline{\triangle H_{dc}} \propto n^{-0.93}$. In a similar fashion we compute from Eqns. (2.1.4) and (2.1.6) the average spacing between dips $\overline{\triangle h_{dc}}$, plotted in Figure 4.1.7(b) for several values of the standard deviation σ of the Gaussian distribution. We find that the slope converges to -0.98 for $\sigma \geq 2$. Using Eqns. (2.1.4) and (2.1.9), we also find a very good linear fit for $\overline{\triangle h_{dc}}$ versus n with slope -0.97. So both calculation and experiment suggest that as n increases, the slope asymptotically approaches -1, ie. $\overline{\triangle H_{dc}} \propto n^{-1}$. We thus conclude that the decrease in spacing of the dips with n in Figure 4.1.5 can be semiquantitatively understood by the zero-order model, and that it is not sensitively dependent on the assumed distribution function $F(A)$, other than that it should monotonically decrease for large loop areas

A; however, the area distribution function Eqn. (2.1.9) fits the data better than Eqn. (2.1.6) for small n.

Figure 4.1.8 shows $P(nf)$ versus h_{dc} computed from the first-order model, Eqn. (2.1.13). This model provides a mechanism for dissipation, and creates both the inductive and dissipative phases for each harmonic, so it is not surprising that the dips are broader and less resolved, e.g., $n = 15$ and 30, because of the interference between the two phases. In other words, the Fourier transform of $\langle V_n(t) \rangle$ now contains both real and imaginary components, and both must vanish to give a deep power dip. The pattern is more complex, and, in fact, this feature is qualitatively observed, e.g., in Figure 4.1.5(f), if the hysteresis is ignored. In principle this model is superior to the zero-order model, but it was unfortunately not evaluated in detail owing to the long times required for the computation (roughly a factor of 10^3 greater than the zero-order model).

Why does a random sample show pseudo "flux quantization"? Recognizing that various versions of the superconducting loop model can explain the experimental finding of deep dips in the harmonic power, almost periodic in the dc field, an interesting question can be asked: How does it come about mathematically that all this structure is not averaged out in, say, Eqn. (2.1.4). Or, to put the question in physical terms, why does a random powder sample of $YBa_2Cu_3O_7$ show sharp dips, eg., as in Figure 4.1.5(e), quite similar to those observed in fabricated thin film arrays of superconducting wires, or arrays of *identical* Josephson junctions? It will be easier to first answer this question mathematically by examining Eqn. (2.1.4).

The general behavior of the harmonic voltage signal, as modeled by Eqn. (2.1.4), is clearly determined by the integral in the equation. To understand the structure of the signal as a function of h_{dc}, one can separate the integrand, say that for odd n, into two factors: the periodic factor $\cos(Ah_{dc})$ and the amplitude $Q_n(h_1, A) \equiv A\, J_n(Ah_1)\, F(A)$. We have neglected the average over orientations; so one can interpret A as the dimensionless *projection* area. One recognizes that $Q_n(h_1, A)$ is, within a

constant factor, the Fourier cosine transform of $\langle V_n(h_{dc})\rangle$, for odd n; for even n, it is the sine transform. For $n \geq 1$, the Bessel function $J_n(x)$ initially increases rapidly as $J_n(x) \approx (x/2)^n/(n!)$, and then behaves like a damped oscillation. If the area distribution $F(A)$ is a sufficiently rapidly decreasing function, at least for large enough values of A, then $Q_n(h_1, A)$ will be a *rapidly decreasing oscillating function of A, with a well-defined peak at* A^*_{n,h_1} , and with the value of A^*_{n,h_1} dependent on n and h_1.

For example, in Figure 4.1.9(a), we have used $F(A)$ from Eqn. (2.1.9) and computed $|Q_n(h_1, A)|$ versus A for the parameters $n = 10$ and $h_1 = 5.0$. It indeed shows successive decreasing peaks with the dominant peak at $A = A^*_{n,h_1} = 2.09$, larger by a factor of 6.7 than the next peak. Thus the integral could *roughly* be evaluated at only the dominant value of A:

$$\langle V_{n,h_1}\rangle \sim A^*_{n,h_1} J_n(A^*_{n,h_1} h_1) F(A^*_{n,h_1}) \sin(A^*_{n,h_1} h_{dc}) \qquad (4.1.3)$$

giving a harmonic power $P_n(h_{dc})$ with h_{dc}-dependence roughly in the form of $\sin^2\left(A^*_{n,h_1} h_{dc}\right)$, and hence with periodic spacing between dips $\triangle h_{dc} \approx \pi / A^*_{n,h_1} = 1.50$, in good agreement with the directly calculated value of $\overline{\triangle h_{dc}} = 1.52$ using the full expression Eq. (2.1.4). To show the dependence on n, we plot, in Figure 4.1.9(b), $|Q_n(h_1, A)|$ versus A for $n = 5$, $h_1 = 5.0$, also using Eq. (2.1.9) for $F(A)$. In this case the location of the dominant peak decreases approximately by a factor of 2, to $A^*_{n,h_1} = 1.10$, corresponding to dip spacing of $\triangle h_{dc} \approx \pi/A^*_{n,h_1} = 2.86$, again in good agreement with $\overline{\triangle h_{dc}} = 2.80$, directly calculated from Eq. (2.1.4). Additional computation shows that A^*_{n,h_1} is approximately proportional to n, in agreement with the full integral and also with the data, where $\overline{\triangle H_n} \propto n^{-1}$, as in Figure 4.1.7, for large enough n.

One can take the view that A^*_{n,h_1} is an "effective loop area" in the sense that the dip spacing $\triangle h_{dc} \propto \left(A^*_{n,h_1}\right)^{-1} \propto n^{-1}$ is, for large n, determined by the larger areas A in the distribution and for small n by the small areas. In some sense $Q_n\left(h_1, A^*_{n,h_1}\right)$ is a "sensitivity factor": out of the wide distribution of supercurrent loop areas, observation

of the n^{th} harmonic selects out only areas near A^*_{n,h_1}. In other words, it means that in an ensemble of superconducting loops with a distribution of loop areas, for a given ac magnetic field amplitude, each generated harmonic is dominantly contributed by loops within a relatively narrow range of loop areas, thus manifesting their flux quantization phenomena in the harmonic.

To examine the dependence of $\triangle h_{dc}$ on the ac field h_1, we show in Figure 4.1.9(c), $|Q_n(h_1, A)|$ versus A for $n = 5$, $h_1 = 10.0$. The dominant peak of $|Q_n(h_1, A)|$ is now located at $A^*_{n,h_1} = 0.58$, corresponding to dip spacing $\triangle h_{dc} = 5.45$, in good agreement with that computed directly from Eq. (2.1.4), $\overline{\triangle h_{dc}} = 5.39$. Additional calculation shows that approximately $\triangle h_{dc} \propto h_1$, i.e., $\triangle H_{dc} \propto H_1$, for large enough values of H_1. The overall result is $\triangle H_{dc} \approx 3H_1/n$. We show below that this behavior is observed experimentally.

To summarize, the unexpected observation of sharp, almost periodic dips in the n^{th} harmonic power with dc field for a distribution of loop areas A can be understood semiquantitatively as the consequence of the folding of a decreasing function $F(A)$ and the rapidly increasing part of the Bessel function $J_n(x)$. However, it should be noted that even though the Bessel functions arise from the properties of Josephson junctions, we have not shown that it is necessary as well as sufficient to ascribe the phenomena to Josephson junctions and hence to flux quantization.

Structure in the intermediate-H_1 region. Figure 4.1.10 shows what happens to $P(16f)$ versus H_{dc} as the ac field is reduced from $H_1 = 23$ to 2 Oe. The spacing $\triangle H_{dc}$ decreases, initially linearly with H_1, as expected from the argument in the previous paragraph. Then structure develops, which seems irregular, and depends on the sense of the H_{dc} scan, e.g., Figures 4.1.10(c) and (e). However, if the leftward trace is reversed, shown as the dotted line in Figure 4.1.10(d), it superposes exactly on the rightward trace. This is the same property shown by the traces in Figure 4.1.5 and is possibly a consequence of fluxon pinning and depinning, but here the narrow spacing gives an

enhanced effect. In Figures 4.1.10(i) and (j) there are many resolved and reproducible sharp dips not uniformly spaced but with average spacing still roughly proportional to H_1. This behavior in the intermediate-H_1 region is similar to the flux jumps observed in nonresonant microwave absorption in low fields.

A set of $P(nf)$ versus H_{dc} traces taken as in Figure 4.1.10 but for *odd* harmonics shows similar behavior for large H_1, with $\triangle H_{dc}$ linearly proportional to H_1, but with the dips decreasing in amplitude as H_1 becomes small. The sharp spikes in Figures 4.1.10(i) and (j) are not observed for odd harmonics.

Figure 4.1.11 summarizes the experimental results in a plot of the values of H_{dc} for the dips versus the magnitude of H_1. The circles denote positions of well-resolved dips, the diamonds regions of unresolved and more closely spaced dips. The solid lines are plots of the dip positions h_{dc} versus the ac field h_1, computed for the zero-order model, Eqs. (2.1.4) and (2.1.9). This figure shows graphically the general agreement between experiment and theory for large H_1, with poor agreement for $H_1 < 5$ Oe.

Structure in the second harmonic. The smooth deep narrow dip in $P(2f)$ versus H_{dc}, Figure 4.1.4, are observed only in the moderately high-H_1 region ($H_1 > 2$ Oe). Figure 4.1.12 shows the behavior in the same sample as the ac field is reduced to the intermediate region: at $H_1 = 1$ Oe the dip has broadened; at $H_1 = 0.5$ Oe there is a fairly abrupt transition to a wide dip with more hysteresis; at 0.4 the pattern is seen to consist of three dips, which broaden and change shape at 0.2 Oe. Although this structure is not predicted by the zero-order model, we find the loop model does predict similar behavior, shown in Figure 4.1.13, computed from Eqs. (2.1.12) and (2.1.9). Although this model does not predict the hysteresis, whose origin is noted above, the abrupt onset of the structure is reasonably related to the experimental behavior. However, this phenomenon has not been explored in detail in other samples.

4.2 Measurements and models on a Ceramic Y-Ba-Cu-O Cylinder

In this section, experimental data taken on a bulk ceramic $YBa_2Cu_3O_7$ cylinder are presented. In this case, extensive harmonics are also generated when the sample is driven by an ac magnetic field. Like in the case of powdered $YBa_2Cu_3O_7$, the harmonics show the same symmetry with respect to $H_{dc} = 0$; when plotted versus the dc magnetic field superposing on the ac field, even harmonics show a sharp dip in power at $H_{dc} = 0$, while the odd harmonics have a peak in power. Also, as in the case of powdered $YBa_2Cu_3O_7$, there are modulations in harmonic power when the dc magnetic field is varied, and dips and peaks in $P_n(H_{dc})$ are observed for both even and odd harmonics.

However, there are important qualitative differences between the harmonic data of bulk ceramic cylinder and the powdered sample. The most obvious one is that when the harmonics are plotted versus dc magnetic field, the dips in power are no longer as regularly spaced and as sharp as in the case of powdered $YBa_2Cu_3O_7$. We found that there is no way to fit the harmonic data of the bulk ceramic $YBa_2Cu_3O_7$ cylinder with the zero-order model. Another obvious question one has to ask oneself in the present experiment is where the supercurrent, which is giving rise to these harmonics, is flowing. For the powdered sample case, there is no doubt that the currents are flowing within the physical grains of the powder, with the "grains" themselves possibly composed of subgrains or cracks that give rise to the rf-SQUID-like behavior. But in the case of a bulk ceramic, one has to consider also the possibility that large supercurrents may be flowing between the grains and throughout the bulk volume of the sample. In fact, the ceramic superconductors have been suggested by many papers in the literature to be "natural" 3-dimensional superconducting array systems formed by highly superconducting grains coupled by Josephson junctions [54][55][56][57]. They are "natural" in the sense that they are not intentionally designed and fabricated arrays such as those 2-dimensional regular arrays of low-temperature superconducting wires

and Josephson junctions which had been fabricated and experimented on even before the high-temperature superconducting materials were discovered [58][59][60]. Instead, the 3-dimensional arrays as suggested are formed naturally during the process of high compression and sintering in the making the earliest high-T_c materials, before single-crystals and thin-films became available.

This suggestion that the high-T_c ceramics are 3-dimensional superconducting arrays also brings up the conjecture that the ceramic should also behave as a "gauge-glass," which is in a sense the superconductivity version of spin-glasses. In this picture, the individual highly superconducting grains are coupled by Josephson effects to give phase coherence and hence superconductivity to the overall, macroscopic system. However, at high enough temperatures thermal fluctuations may be strong enough to disrupt the coupling, and the phases of the individual grains' pairing order parameters will fail to remain coherent over the whole sample. So, at these temperatures, which are between the intrinsic critical temperature of the intragranular material and the 3-dimensional array's phase-locking temperature, the sample as a whole is not superconducting, even though Cooper pairs exist within individual grains. Monte-Carlo simulations have suggested that at the phase-locking temperature of the overall array, there is indeed a real phase transition occurring [52].

However, physically reasonable as these suggestions are, there has not been, to my knowledge, any unambiguous experimental evidence about the *coexistence* of the intergranular Josephson supercurrent in the ceramic $YBa_2Cu_3O_7$ together with the intrinsic supercurrents inside the individual grains. The question of the existence of the 3-dimensional superconducting array's phase-locking temperature in the high-T_c ceramic has also been rarely addressed experimentally. In the present section, one of our objectives is to present *unambiguous* experimental evidence for the existence of both the intergranular Josephson supercurrent and the intrinsic intragranular supercurrents within the same piece of ceramic sample. Another objective is to show that the generalized

critical state model presented in Chapter 2 describes the intergranular superconducting medium of the ceramic quantitatively. Lastly, experimental measurement of the phase-locking temperature of the 3-dimensional array formed by the superconducting grains, as separate from the intrinsic critical temperature of the grains, will be presented.

The bulk ceramic $YBa_2Cu_3O_7$ sample that we used in this experiment is in the form of a cylinder (sample no.: C-46N), 2.3 cm long and 3.1 mm diameter. When this sample is cooled in liquid nitrogen in zero magnetic field, extensive harmonic components picked up by the receiver coil can be detected by the spectrum analyzer. In this case, the receiver coil, which is made up of gauge 40 wire, is wound tightly and directly on the sample itself, covering the middle of the sample's length (78 turns in two layers; cross-sectional area = .074 cm^2).

Harmonic power versus H_{dc}; measurements and initial modeling. As mentioned above, the harmonics generated by the bulk ceramic sample show the same symmetry with respect to H_{dc} as the powder samples — even harmonics have a dip in power at $H_{dc} = 0$, while odd harmonics have a peak. In Figure 4.2.1(a), harmonic power plotted as a function of dc magnetic field superposed on the ac field is shown for harmonic numbers n = 2, 5, 6 and 10. The ac magnetic field amplitude H_1 is 13.5 Oe in this case, while the temperature is 77 K. The arrows indicate the direction in which the dc magnetic field is scanned. Figure 4.2.1(c) shows data at $H_1 = 4.5$ Oe.

Because of the bulk nature of the sample, in an attempt to explain the data we ignore for the moment the granular nature of the sample and assume that supercurrent is flowing throughout the sample as if it were a continuous medium, as in Section 2.2. Since the materials are known to be type-II superconductors, the modified critical-state model would seem appropriate to describe the system. In fact, using Eqns. (2.2.9) and (2.2.10) with $\beta = 1.8$ and $H_0 = 3.0$ Oe, the model calculation fits the harmonic power versus H_{dc} data very well up to the 10^{th} harmonic; these values of β and H_0 were empirically determined by model fitting. Beyond the 10^{th} harmonic the calculation no

longer gives a very satisfactory fit (not shown). In Figures 4.2.1(b) and (d), these model calculations are compared side by side with the data, with the same corresponding ac field amplitude H_1, harmonic numbers n and dc magnetic field range H_{dc}. The fit is quite good. To highlight this significant improvement in data-fitting by the introduction of the parameter β to the original Kim-Anderson model, the model calculation of the 10^{th} harmonic power with $H_1 = 4.5$ Oe is repeated using $\beta = 1.0$, in accordance with the Kim-Anderson model, and is plotted as the dotted line in Figure 4.2.1(d). One can readily see that the original model calculation fails to explain crucial features of the data.

One can observe that the Kim-Anderson model, and its modification, does predict the observed symmetry of $P(nf)$ with H_{dc}, as pointed out by Ji *et al* [8]: the addition of a dc field breaks the symmetry, in a mathematically analogous manner to the symmetry breaking in the Josephson junction models of Section 2.1.

From the data-fitting parameters, we deduced that the critcal current density J_c of the supercurrent is equal to 790 A/cm^2 at zero magnetic field, and it drops to 80 A/cm^2 by the application of only 7 Oe of magnetic field. Both the low value of the critical current density J_c and its sensitivity to low magnetic fields indicate that the supercurrent that we are dealing with in these harmonic data is most likely the intergranular Josephson current. This is consistent with our earlier assumption that the supercurrent mainly responsible for the present harmonic data is flowing throughout the sample as if the sample were a continuous medium.

However, if the harmonic data here are really mainly contributed by the intergranular supercurrent, then one has to ask what the contributions, if any, from the intragranular supercurrents are. As a matter of fact, one can even ask if there really are supercurrents flowing within the grains; is there any solid evidence of their existence inside a bulk ceramic sample when the grains are all coupled together?

Observation of both inter- and intragranular components of the complex permeability.

Our next objective is to find unambiguous experimental evidence for the coexistence of both the inter- and intragranular supercurrent components, and to measure the corresponding critical current densities J_c, inside the same ceramic $YBa_2Cu_3O_7$ sample. In order to do this, we take advantage of the knowledge that, in the critical state model, when the superposing dc magnetic field is zero, the value of μ_1'', which is proportional to the ac hysteretic absorption by the sample normalized by the square of the ac field amplitude H_1^2, is at a maximum if $H_1 = H^*$, where H^* is the value of the external field at which the penetrated flux front reaches the center of the sample (see Section 2.2 and Appendix A). As described in Eqn. (2.2.11), the value H^* increases as a function of increasing critical current density J_c as

$$H^* = H_0 \left\{ \left[1 + \frac{4\pi(\beta+1)}{cH_0} J_{c0} R \right]^{\frac{1}{\beta+1}} - 1 \right\}, \qquad (4.2.1)$$

where $J_{c0} \equiv J_c(H=0)$. To see through this expression for H^*, one can take $\beta = 0$, as in the original Bean version of the critical-state model, and find that $H^* = 4\pi J_c R / c$. The idea is that if intragranular supercurrent exists in the ceramic sample, its critical current density $J_{c,g}$ should be of about the same order of magnitude as those measured in single crystals and thin films, ie. $J_{c,g} \sim 10^5 - 10^7$ A/cm^2. This value is many orders of magnitude larger than the critical current density, presumed at this point to be intergranular $J_{c,J}$, that we deduced above, namely 790 A/cm^2 at $H = 0$. Thus if one applies the critical-state model separately to the macroscopic bulk ceramic sample and to the individual grains, the values of H^*_{inter} and H^*_{intra} should be very different in magnitude. So when the ac absorption is measured as a function of ac field amplitude H_1, with H_1 covering a large enough range, the two normalized absorption peaks at H^*_{inter} and H^*_{intra}, if they exist, should be detected.

Experimentally this is achieved by noting that from Eqn. (2.2.21) for $n = 1$ one can measure with a lock-in detector an in-phase signal voltage $V_1' \propto \mu_1' H_1$, and an out-of-phase signal voltage $V_1'' \propto \mu_1'' H_1$. In Figure 4.2.2(a), we plot as circles the

experimentally measured μ_1'' versus $\log_{10}(H_1)$ over a wide range $0.1 < H_1 \leq 400$ Oe at $f = 85$ Hz, $T = 77$ K, $H_{dc} = 0$. In this case, two μ_1'' peaks are observed, the lower one at $H_1 = 15$ Oe and the much higher one at $H_1 = 250$ Oe, which we ascribe to the inter- and intragranular contributions, respectively. The lower peak was found to be independent of frequency in the range $85 - 10^4$ Hz, but apparatus limitations of $|H_1|$ at high frequencies did not allow a similar conclusion for the high-field peak. Figure 4.2.2(b) is a plot of the inductive component μ' of the fundamental mode complex permeability versus $\log_{10}(H_1)$ which also shows, but less distinctly, two components: (i) the flat plateau region $15 \leq H_1 \leq 80$ Oe, due to full vortex penetration of the intergranular medium, corresponds to $\mu' = 0.17$ which is the effective permeability μ_{eff} for this medium at 77 K; (ii) a steeply rising second region which does not reach a plateau at this temperature owing to insufficient H_1 field availability.

To find out for sure which μ_1'' peak is really due to which supercurrent component, the same $YBa_2Cu_3O_7$ sintered bar material was ground in an agate mortar to a fine powder (sample No. C-48A); optical microscope examination showed grains of sizes $1 \leq R_g \leq 60$ μm, with *rough* average $\overline{R_g} \sim 10$ μm. To further isolate the grains from one another, one volume of this powder was mixed with one volume of 1 μm grit plus one volume of 0.1 μm grit Al_2O_3 powder (sample No. C-48B). Both powder samples display similar behavior. The data in Figure 4.2.2(c) are for sample C-48B. Here, the dissipative component μ_1'' of the complex permeability is plotted versus $\log_{10}(H_1)$, showing essentially no evidence for the previous peak at low H_1 field, while the high-field peak at $H_1 = 250$ Oe remains. This high-field peak is therefore ascribed to the collective contributions of the intragranular supercurrents within individual grains, and the low-field peak at $H_1 = 15$ Oe of the ceramic sample is ascribed to the intergranular Josephson current. We conclude that powdered samples do not allow significant circular shielding currents on the scale of R, but only on the scale of R_g, thus invalidating the above critical-state model for the intergranular medium.

The measurements of H^*_{intra} and H^*_{inter} also enable one to get a quick estimate of the intra- and intergranular critical current densities respectively. For the intragranular current, taking the rough estimate of $\overline{R_g}$ to be 10 μm as above and using the simple Bean version of the critical-state model, ie. J_c taken to be independent of field, one gets from the expression $J_{c,g} = c\,H^*_{intra} / 4\pi\,\overline{R_g} = 2 \times 10^5$ A/cm^2. However, using the generalized critical state model with fitting parameters $\beta_g = 1$, $H_{0,g} = 5$ Oe and $H^*_{intra} = 250$ Oe, as discussed later in this Section, one gets $J_{c,g}(H = 0) = 5 \times 10^6$ A/cm^2. Similarly, using the same method, one gets an estimate of 78 A/cm^2 for $J_{c,J}$. The latter estimate agrees with the value 80 A/cm^2 of $J_c(H = 7$ Oe) deduced earlier from fitting the high harmonic power versus dc magnetic field data with the generalized critical-state model using $\beta = 1.8$ and $H_0 = 3$. This should be expected; the Bean version assumes J_c to be independent of H, and hence the flux density profile to be linear as a function of depth into the sample, it should only give an estimate of $J_c(H)$ at $H \sim 0.5\,H^*$. As mentioned earlier, model calculations yields for the intergranular component $J_c(H = 0) = 790$ A/cm^2.

Harmonic power versus ac magnetic field measurements. Similarly clear distinction of the inter- and intragranular supercurrents in the ceramic sample is also manifested in the harmonics. In Figure 4.2.3(a), with $H_{dc} = 0$, the third harmonic power (in dB) generated by the ceramic sample is measured as a function of $\log_{10}(H_1)$ for almost five orders of magnitude and is plotted as circles. These data were obtained by plotting the video readout (dB) of the HP spectrum analyzer : $10\log_{10}\left[(V'_n)^2 + (V''_n)^2\right] = 10\log_{10}\left[(\mu'_n)^2 + (\mu''_n)^2\right] + 10\log_{10}\left[H_1^2\right] + constant$, which does not measure separately μ'_n and μ''_n. When this plot is compared to similar data taken on the powdered sample C-48B in Figure 4.2.3(b), it can be readily seen that the steep slope of $P(3f)$ for $H_1 \geq 100$ Oe is mostly due to the intragranular supercurrents, whereas the "hump" located in the range $0.4 \leq H_1 \leq 100$ Oe, and broadly peaked at $H_1 \approx 25$ Oe is due to the intergranular supercurrents.

Similar features can also be observed for higher harmonic data. In Figures 4.2.3(c) and (d) the fifth harmonic power data taken at zero dc field on the ceramic and powdered samples respectively are plotted as a function of $\log_{10}(H_1)$ as circles. Again, the steep slope of the ceramic data for $H_1 \geq 100$ Oe is due to the intragranular supercurrents, while the "hump" at $H_1 < 100$ Oe is due to the intergranular current. For completeness, the seventh and the ninth harmonic power data taken at zero dc field on the ceramic are also plotted as a function of $\log_{10}(H_1)$ in Figures 4.2.4(a) and (b), respectively. The modeling of $P(nf)$ versus $\log_{10}(H_1)$ will be presented after the μ_1', μ_1'' data are discussed.

Modeling μ_1' and μ_1'' versus $\log_{10}(H_1)$. Now that we have distinguished experimentally the separate contributions of the inter- and intragranular supercurrents to the fundamental mode complex permeability versus $\log_{10}(H_1)$ as well as harmonic power versus $\log_{10}(H_1)$ data, it is desirable to ask if the fitting parameters ($H_0 = 3$ Oe, $\beta = 1.8$, $H^* = 15$ Oe) acquired earlier by means of model fitting the high harmonic power versus dc field data are consistent with the $\log_{10}(H_1)$ data. As asserted above, the earlier $P(nf)$ versus H_{dc} data should be due mainly to the intergranular supercurrent. Since in the fundamental mode complex permeability $\tilde{\mu}$ and $P(nf)$ versus $\log_{10}(H_1)$ data, the inter- and intragranular components are so well distinguished, we are now in a unique position to test the assertion.

In Figure 4.2.2(d), the circles are measurements of μ_1'' as a function of $\log_{10}(H_1)$, with a superposing dc magnetic field of 10.2 Oe. The low-field μ_1'' peak splits into two, one at about 7.5 Oe and the other at about 25 Oe. While the latter field value (25 Oe) is equal to $H^* + H_{dc}$, the former (7.5 Oe) is approximately equal to the ac field amplitude at which the *ac* flux front reaches the axis at $H_{dc} = 10.2$ Oe, ie. when the right-hand-side expression of Eqn. (A.2.1) (see Appendix A) goes to zero. In fact, this splitting of the low-field peak is fit very well quantitatively by the generalized critical-state model using the same model parameters as used above, namely $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$

Oe. The solid lines in Figure 4.2.2 represents these model calculation results for the intergranular component. The dashed lines in Figure 4.2.2 are model calculations for the intragranular component, using $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe, fitting the data semi-quantitatively.

As it turns out, applying a small H_{dc} to split the μ_1'' peak provides a more convenient way for model fitting. The heights of the split peaks, together with their positions are good criteria for determining the best fitting parameters β and H_0 . This is less time-consuming than fitting the previous $P(nf)$ versus H_{dc} data, although those data provide valuable cross-checking.

As the dc magnetic field gets higher, both the inter- and intragranular μ_1'' peak positions shift to lower H_1 values. This reflects the decrease of both the inter- and intragranular supercurrents as a function of increasing magnetic fields. In Figures 4.2.5 to 4.2.8, the measured dissipative and inductive components of complex permeability $\tilde{\mu}_1$ are plotted as a function of $\log_{10}(H_1)$, with superposing dc magnetic field $H_{dc} =$ 51.4, 60.3, 70.2 and 90.4 Oe. As indicated by the low-field μ_1'' peak, the intergranular current is sustained in a relatively high dc magnetic field of 90.4 Oe. The intragranular absorption peak surprisingly increases in amplitude with the application of dc magnetic field. This is not explained by the generalized critical-state model for a wide range of H_0 and β values. In fact, the generalized critical-state model can only give semi-quantitative fit (see the dashed lines) to the intragranular component of *all* the data. This is not understood at present.

Modeling $P(nf)$ versus $\log_{10}(H_1)$. Having first discussed the $\tilde{\mu}$ versus $\log_{10}(H_1)$ data we now return to the power data for the C-46N sample, Figures 4.2.3 and 4.2.4, for which $H_{dc} = 0$. The solid lines are model calculations for the intergranular components, using the same set of parameters as above: $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. The dot-dashed line is the model calculation for the intragranular component, using parameters $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe, while the dashed line is a

similar calculation using $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe, plotted for comparison. We see that for $n = 3$, 5, 7 and 9, this set of model parameters gives a surprisingly good fit to the data over 5 decades of H_1 and about 10 decades of $P(nf)$, especially for the intergranular component. In particular the broad saturation of $P(nf)$ versus $\log_{10}(H_1)$ at $H_1 > 20$ Oe is well explained, as is the rapid rise at $H_1 \approx 10$ Oe for $n = 7$, 9. We note that for $H_1 \leq 1$ Oe the data are limited by the apparatus sensitivity.

Harmonic power versus $\log_{10}(H_1)$ data also show up interesting features with the application of dc magnetic fields which can provide further cross-checking of our model calculation fits. In Figure 4.2.9(a), $P(7f)$ taken on the ceramic sample C-46N in a dc field of 10.1 Oe is measured as a function of $\log_{10}(H_1)$. This is to be compared to Figure 4.2.4(a), for $H_{dc} = 0$. The circles are experimental data, while the solid line is the generalized critical-state model calculation for the intergranular component, with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. The extra features in the intergranular "hump" introduced experimentally by the small dc field are remarkably well explained by the model calculation. The dip in $P(7f)$ at about $H_1 = 120$ Oe is probably formed by out-of-phase cancelling of the inter- and intragranular contributions.

The circles in Figures 4.2.9(b), (c) and (d) show $P(5f)$ data taken on the sample C-46N versus $\log_{10}(H_1)$ in dc fields $H_{dc} = 30$, 60 and 90 Oe; the lines are corresponding model calculation results similar to those described above.

Measurements and modeling of $P(nf)$ versus frequency. Another severe test of the generalized critical state model is the measurement of the harmonic power spectra of the signal generated by the ceramic sample, C-46N. In Figure 4.2.10, such spectra are measured at $H_{dc} = 0$ for $H_1 = 5$, 10, 20 and 40 Oe, and are shown as solid lines. The fundamental driving frequency is 1000 Hz and the temperature of the sample is at 78.5 K. The squares are the theoretical spectra, normalized to the experimental $P(3f)$, predicted by model calculations for the *intergranular component only*, using $\beta = 1.8$, $H_0 = 3$ Oe and the experimentally measured value of $H^* = 13$ Oe at 78.5 K. The good

fits indicate again that up to $H_1 = 40$ Oe, the dominant contributions to the harmonic power are due to the intergranular supercurrent.

Temperature dependence of the inter- and intragranular components of the complex permeability: two transition temperatures. Now that we can detect the inter- and intragranular supercurrent components separately by measuring the complex permeability versus $\log_{10}(H_1)$, it is interesting to monitor their behavior as a function of temperature.

Figures 4.2.11 to 4.2.15 show for C-46N and $H_{dc} = 0$ measured values of μ_1' and μ_1'', plotted versus $\log_{10}(H_1)$ over about five decades at 78.8, 83.7, 86.0, 87.5, and 89.5 K. The peak positions of μ_1'', corresponding to H^*_{inter} and H^*_{intra}, both decrease as the temperature increases. This is due to the decrease of both the inter- and intragranular cr tical current densities as the critical temperature is approached. Another feature to notice is the increase of the effective permeability of the intergranular medium of the ceramic as a function of temperature. This effective permeability is determined by the plateau in the μ_1' versus $\log_{10}(H_1)$ plots for $H^*_{inter} < H_1 < H_{c1,g}$. In Figure 4.2.16 are plotted μ_{eff} versus temperature. This increase is due to the fact that as the temperature rises, the London penetration depth of the superconducting grains become larger, thus increasing the portion of the sample volume into which flux can penetrate without the magnetic field exceeding the lower critical field $H_{c1,g}$ of the grains. At sufficiently low temperature, the London penetration depth is much smaller than the grain sizes: $\lambda_g \ll \overline{R_g}$. In this limit, Eqn. (2.2.3) in Chapter 2 will become $\mu_{eff} \approx f_n$, where f_n is the volume fraction of the normal region. If one simple-mindedly ignores the demagnetization effects of the grains also, the *lowest order* correction of μ_{eff} in powers of $\lambda_g / \overline{R_g}$ is:

$$\mu_{eff} \approx f_n + \left[2 f_s \lambda_g(T) / \overline{R_g} \right] . \qquad (4.2.2)$$

From this expression, one can understand how the increase of the London penetration depth can increase μ_{eff}. The dashed line in Figure 4.2.16 is the calculation of Eqn.

(4.2.2), using $f_n = 0.144$, $\lambda_g(T=0) = 1400$ Å [18], $T_c = 91.2$ K (see Figure 4.2.17 below), $\overline{R_g} = 10$ μm and the empirical "two-fluid" approximation of the temperature-dependent penetration depth [61]:

$$\frac{\lambda_g(T)}{\lambda_g(0)} = \left[1 - (T/T_c)^4\right]^{-0.5} . \tag{4.2.3}$$

Note that the value of f_n has been chosen to be the ratio of the measured density of this sample (5.46 g/cm^3) to the density $\rho = 6.38$ g/cm^3 computed from x-ray structure [12].

When the inter- and intragranular penetration fields, H^*_{inter} and H^*_{intra}, are plotted as a function of temperature, as in Figure 4.2.17, one interesting feature of this sample becomes obvious — the intergranular penetration field H^*_{inter} extrapolates to zero at 4.6 degrees below the temperature at which the intragranular one does. Remembering that the penetration field H^* is a measure of the critical current, this means that the ceramic sample *as a whole* does not go superconducting until at 4.6 degrees below the intrinsic critical temperature of the intragranular superconducting material. The critical current densities are deduced from H^*_{inter} and H^*_{intra}, using the critical state model, and are plotted in Figure 4.2.18; the specific equation used is, from Eqns. (2.2.9) and (2.2.11),

$$J_c(H=0) = \frac{10}{4\pi(\beta+1)\,RH_0^{\beta}}\left[(H^*+H_0)^{\beta+1} - H_0^{\beta+1}\right] , \tag{4.2.4}$$

where J_c is in A/cm^2, R is in cm, and H^*, as well as H_0, are in Oersted. $J_{c,J}(H=0)$ is calculated using parameters $\beta = 1.8$ and $H_0 = 3$, while $J_{c,g}(H=0)$ is calculated using parameters $\beta_g = 1.0$ and $H_{0,g} = 5$.

One explanation of the lower overall transition temperature for the ceramic sample as a whole is that the sample is essentially a random 3-dimensional matrix of highly superconducting grains pressed into contact with one another through Josephson weak links. According to this picture [52][39], each superconducting grain acquires a gap, or pairing order parameter, as the temperature is lowered below the single-grain transition temperature T_{c0}. The amplitude of this order parameter $|\psi|$ is fixed by the characteristics

of the single grain, so is the intragranular condensation energy $E_g = (H_{cg}^2/8\pi)\,V_g$, which is proportional to $|\psi|^2$ to the lowest order correction in $|\psi|$. However, the phase of the order parameter is not fixed if there were no intergrain Josephson coupling, which is characterized by the Josephson coupling energy $E_J = (h/8\pi e)\,I_0$, where I_0 is the maximum Josephson current I_0. The order parameter thus behaves as a two-component ("x-y") spin and the matrix of grains may be represented by a set of vectors in the complex plane. The weak Josephson coupling between grains acts like a ferromagnetic interaction between the "spins" in the absence of an applied magnetic field. The phases of the grains are thus "locked" by the coupling, phase coherence across the whole junction matrix is established, and the whole ceramic sample becomes superconducting.

However, thermal fluctuations become significant when $k_B T \gg E_J$, which is proportional to $(T_{c0} - T)$. In this temperature range thermally activated phase slippage readily occurs in the junctions and so a time-averaged voltage appears across any junction that carries current. The sample as a whole becomes resistive, even though all individual grains may still be strongly superconducting. This state has been referred to as a "paracoherent" state by some authors on granular *low-temperature* superconductors [62][63][64]. This picture of granular superconductors is analogous to the superparamagnetic-ferromagnetic transition in granular ferromagnetic thin films [65][66]. The cross-over between phase-locked (coherent) and phase-fluctuation-dominated (paracoherent) behavior occurs at a Josephson phase-locking temperature T_{cJ} given roughly by the equation $E_J(T_{cJ}) \approx k_B T_{cJ}$. In Ref. [39], it is derived that

$$T_{c0} - T_{cJ} \approx (1.57 \times 10^{-8}\ \mathrm{A/K})\ T_c^2 / I_0(0)\ . \qquad (4.2.5)$$

Simple-mindedly assuming $I_0(0) \approx J_{cJ}\,a^2$ and taking $J_{cJ} \approx 700\,\mathrm{A/cm^2}$, and $a \approx 2\,\mu\mathrm{m}$, and $T_{c0} = 92$ K, one gets $T_{c0} - T_{cJ} = 4.7$ K. More theoretical studies on disordered arrays of weakly-coupled superconducting grains have been done. For example, Monte Carlo simulations done by Shih, Ebner and Stroud [52] predicted that in such a random network,

there is indeed a thermodynamic superconducting phase transition for the overall system at the phase-locking temperature. Relevant experimental works have also been done on related systems of conventional superconductors. In 2-dimensional networks of niobium Josephson junctions, it was found that the superconducting transition temperature of the network is lower than that of the niobium islands. In essence, according to this explanation, the lower intergranular transition temperature in Figures 4.2.17 and 4.2.18 is the phase-locking temperature of the ceramic sample. Between this and the intrinsic critical temperature of Y-Ba-Cu-O, the individual grains in the sample are superconducting with pairing order parameter of non-zero amplitude. However, due to thermal fluctuations, the phases of the order parameters of the grains are not coherent, so the ceramic as a whole is not superconducting. However, below the phase-locking temperature, phase coherence exist among the grains and the whole ceramic becomes superconducting.

Another picture for the lower intergranular temperature is in terms of the flux-creep picture. In the discussion of the critical state model in Chapter 2, we have omitted the possibility of thermal effects for simplicity. Actually Anderson has managed to explain a lot of phenomena in conventional type-II superconductors by his flux-creep theory.

In the flux creep theory [35][34][67], the pinning centers in type-II superconductors trap Abrikosov flux lines by means of potential wells. By means of thermal activation and aided by the Lorentz force $\mathbf{J} \times \mathbf{B}/c$, the flux lines hop over free energy barriers in the form of bundles.

In the absence of this thermally activated motion, the flux lines are held in free energy potential wells of depth U and width w. The Lorentz force on a flux bundle of volume V is JBV/c. Hence the potential well is reduced to an effective height $(U - JBVw/c)$ by the Lorentz force. If there is no thermal activation, flux flow will occur when $J = J_{c0} = cU/BVw$ as discussed in Chapter 2. The effective well depth can then be written as $U\ (1 - J/J_{c0})$.

However, in the presence of thermal activation, the net diffusion velocity on a flux line is $\Omega d \exp[-U(1 - J/J_{c0}) / kT]$, where d is the distance between pinning centers and Ω is the frequency of oscillation. This may be regarded as expressing the probability that a flux line has enough energy to cross the barrier, in which case Ω would be the frequency of flux line oscillation.

From this mean diffusion velocity the electric field is given by

$$E = \frac{B\Omega d}{c} \exp\left[-\frac{U}{kT}\left(1 - \frac{J}{J_{c0}}\right)\right] . \tag{4.2.6}$$

Thus the voltage-current curve is exponential and the critical current determined in an experiment depends on the lowest voltage which can be measured. If this is E_c then

$$J_c = J_{c0}\left[1 - \frac{kT}{U} \ln\left(\frac{B\Omega d}{E_c}\right)\right] . \tag{4.2.7}$$

In other words, the value of the critical current density is depressed by the presence of thermally activated flux creep and will go to zero at a temperature lower than the critical temperature, defined to be the temperature when $J_{c0} \rightarrow 0$ in the absence of thermal activation.

In the author's opinion, these two superficially different pictures may only be two different "languages" describing the same physical process taking place in the system. The facts that they both involve thermal activation and that in the random matrix picture, phase-slippage also involves motion of flux lines highlight the similarity of the two pictures. There may yet be other possible explanations for the intergranular transitior temperature; for example, the vortex-glass model of M. P. A. Fisher's [68][69][70]. However, most of the experiments and theoretical works to date describe the model in terms of a system with a high dc magnetic field of the order of teslas applied to it, which is not the case in our experiments. Therefore the direct relevance of the vortex-glass model to this case is not obvious.

4.3 Measurements and Model on Y-Ba-Cu-O Thin-film

From the previous section, we have seen that by measuring the inductive and dissipative components of the fundamental mode ($n = 1$) complex permeability as a function of ac magnetic field amplitude H_1 over a large enough range, one can distinguish very clearly the inter- and intragranular supercurrents inside a bulk ceramic Y-Ba-Cu-O sample. From the peak positions of μ_1'' in H_1, called H^*_{inter} and H^*_{intra} respectively in Section 4.2, one can even get a quick estimate of both the inter- and intragranular critical current d[illegible]ies. In this section, similar measurements are reported on a pulsed-laser-ablated Y-Ba-Cu-O thin-film (sample no. C-51) deposited on $SrTiO_3$, with the c-axis oriented perpendicular to the film surface.

The circles in Figure 4.3.1 are the experimentally measured inductive and dissipative components of the complex permeability, μ_1' and μ_1'', of this Y-Ba-Cu-O thin-film shown over four decades of H_1. In these measurements, the 2-coil receiver is used, well balanced with the balancing-circuit described in Chapter 3. Without a superconducting sample, the balanced signal is within 100 ppm of the induced voltage of the individual coils. Only one peak in μ_1'' is observed, located at $H_1 = H^* \approx 95$ Oe. Again, this peak's location is at about the same position as the inflection point of μ_1'. From the fact that μ_1', for H_1 greater than H^*, approaches a plateau at value 1 (corresponding to the normalized in-phase pick-up signal voltage V_1'/H_1 decreasing to zero), no more dissipation peaks are expected beyond the H_1 fields achievable by this apparatus. Thus, this absorption peak at $H_1 \approx 95$ Oe is ascribed to the "one and only intrinsic" supercurrent of the superconducting film.

Can we get an estimate of J_c of this sample, as we did earlier for the ceramic Y-Ba-Cu-O cylinder? This question is a little more complicated due to the geometrical shape of this sample. For the cylinder, the dimensions of the sample (length / diameter ≈ 7) was intentionally chosen to justify, tolerating some error, the neglect of demagnetization,

at least for the intergranular medium. For the disk-shaped thin-film, however, the demagnetization effect is very significant. The large demagnetization factor in this case approaches one: $D \approx 1 - (d/R)$ where d is the thickness and R is the radius of the disk. A tempting approach to deal with this effect would be simply to use the conventional treatment of demagnetization for ellipsoidal samples, and correct the internal field by the demagnetization field — $H_{internal} = H_{applied} - 4\pi DM$, where M is the magnetization of the sample. However, the critical state actually generates a *nonuniform* magnetization through the sample, thus violating the conditions for a conventional treatment. Even taking the average magnetization, assuming it to be uniform and calculating self-consistently the internal field $H_{internal}$ predicts effects which disagree with experimental data [71][72].

Daeumling and Larbalestier [73] treat this demagnetization problem for critical state in disk-shaped superconductors. They divide the disk into ring segments and calculate self-consistently the current distributions and magnetic field vectors due to the combined contributions of the ring-current segments, in addition to the applied magnetic fields. They find that the field shielded (or trapped) in the center of the disk (they call it h^*) is roughly equal to $4\pi J_c d / c$, where d is the *thickness* of the disk! Note that in the absence of demagnetization the shielded (or trapped) field would be $4\pi J_c R / c$ where R is the radius of the disk. The shielding currents also create radial fields which are of order $2\pi J_c d / c$ on the disk surface. For low applied fields $H_{applied} < h^*$ these self-field effects dominate, leading to substantial deviation of the local field from the applied field. They also find that the demagnetization field does not depend on the applied field, but rather on J_c. So if $H_{applied} \gg h^*$, a demagnetization correction is not necessary.

Associating the position of our μ'' peak of the Y-Ba-Cu-O thin-film with h^*, we can still get an estimate of J_c of the film $J_c = 10\, H^* / 4\pi d \approx 1.5 \times 10^6$ A/cm^2, where the film thickness $d \approx 0.5$ μm.

Simple-mindedly adopting the critical state model calculation for a cylindrical

geometry, but merely reinterpreting the penetration field H^* as $\sim J_c d$ rather than $\sim J_c R$, we find that the μ' and μ'' data on the Y-Ba-Cu-O thin film is fit quantitatively by the Bean-version of the critical state model, ie. J_c is taken to be independent of the local field. In Figure 4.3.1, the circles represent data points, whereas the solid lines are the Bean model calculations represented on the same scale as the data.

Experimental harmonic spectra of the signals taken at several H_1's are shown as solid lines in Figures 4.3.2. These spectra are all taken at zero dc magnetic field and 77 K. The squares are the harmonic power spectra, normalized to have the same third harmonic power as the data, calculated for the Bean version critical state model $J_c =$ constant which, for $H_1 < H^*$, can be represented as (please also refer to Eqns. (2.2.16) and Ref. [36]):

$$P(nf) = \left[\frac{5}{(n-2)(n+2)}\right]^2 P(3f) \ . \tag{4.3.1}$$

One sees that the Bean spectrum fits the low H_1 experimental data quite well, and that the fit gradually fails as H_1 gets larger. This is not too surprising since as H_1 gets larger, the effects of the magnetic field dependence, albeit small, of J_c may start to show up.

For completeness, odd harmonic power measurements were taken at 77 K versus $\log_{10}(H_1)$ are shown as circles in Figures 4.3.3 for $n = 3, 5, 7, 9$. These data are taken at zero dc field. However, application of H_{dc} up to 90 Oe does not cause significant change in the data. For example, Figure 4.3.3(c) also shows, as diamonds, a plot of $P(7f)$ versus $\log_{10}(H_1)$ at $H_{dc} = 100$ Oe, showing only little change; the two sets of data lie almost on top of each other, as expected from the Bean assumption that J_c is independent of H.

The solid lines in Figure 4.3.3 are calculations according to the Bean model, with $H^* = 95$ Oe and "R" $= d = 0.5$ μm. The calculations give the correct slopes of the $P(nf)$ versus $\log_{10}(H_1)$ data and also the correct relative amplitudes. However, they could not explain some of the fine details in the data, such as the small "bump"

in the third harmonic in Figure 4.3.3(a). Also, for the 5^{th}, 7^{th} and 9^{th} harmonics, the wiggling behavior occurs at lower applied H_1 values and has smaller amplitudes in the experimental data than in the calculations. Moreover, the amount of displacements both in H_1 and power amplitudes are consistent among all three harmonics. This is not understood at present, and we believe that the geometrical shape of the sample, hence the complicated demagnetization effects involved, is at least one of the major reasons for this effect. We also note that the flat regions in $P(nf)$ for $H_1 \leq 1$ Oe are due to insufficient signal to noise ratio, and represent the background spectrum analyzer noise level at ~ -125 dBm.

As a last remark of this section, we would like to comment on the single peak observed in μ'' versus H_1 data on this sample. This peak, as suggested earlier, is due to the ac absorption of the intrinsic supercurrent in the thin-film. However, it has been suggested in the literature that twin-boundaries in Y-Ba-Cu-O crytals and thin-films may behave like weak-links. So in principle thin-films like ours should also have both inter-(weak) and intragranular (strong) supercurrent components. One possible reason that we have not seen a second ac absorption peak in our μ'' versus H_1 data may be that at 77 K, the different domains of the thin-films are very well coupled across the twin boundaries, and thus the intergranular effects are not prominent enough to be detected. An interesting experiment to try is to measure μ'' versus H_1 at higher temperatures, just as what has been done for ceramic Y-Ba-Cu-O cylinder C-46N in the previous section. At high temperatures, the coupling across twin boundaries may become weaker and the intergranular effect may thus show up in experiments.

4.4 Measurements and Model on Bi-Sr-Ca-Cu-O Single Crystal

In the this section, we present experimental data taken on a $Bi_2Sr_2CaCu_2O_8$ single crystal (sample number C-50). All the data are taken at 77 K with both the ac and dc

magnetic fields perpendicular to the *a-b* plane of the crystal.

The inductive and dissipative components of the complex permeability, μ_1' and μ_1'', of this Bi-Sr-Ca-Cu-O single crystal are measured over four decades of H_1 at 77 K and are plotted as circles in Figure 4.4.1. As in Section 4.3, the 2-coil receiver is used to pick up the signal from the sample. The receiver is well-balanced with the balancing circuit described in Chapter 3. Again, as in the Y-Ba-Cu-O thin-film sample, only one peak in the absorptive component μ_1'' is observed. However, the value of the peak position H^* is much smaller than that of the Y-Ba-Cu-O thin-film: $H^* \approx 5.2$ Oe. Since no more absorption peak is expected beyond the maximum achievable H_1, by the same reasoning as in Section 4.3, this absorption peak at $H_1 = H^* = 5.2$ Oe is ascribed to the intrinsic supercurrent of the superconducting material. Using Daeumling and Larbalestier's results, we can get an estimate of J_c of this single crystal as $J_c = 10H^*/4\pi d \approx 4.1 \times 10^3$ A/cm^2, taking the thickness of the crystal to be 10 μm. This small value of J_c is most likely due to a small flux pinning force density in the sample. The dashed lines in Figure 4.4.1 are calculations according to the generalized critical state model, Eqn. (2.2.10), using $\beta = 1.3$, $H_0 = 3$ Oe and $H^* = 5.2$ Oe. The fits are satisfactory but definitely not as good as those for the ceramic Y-Ba-Cu-O cylinder in Section 4.2 and Y-Ba-Cu-O thin-film in Section 4.3. These fits are obtained by simple-mindedly using the generalized critical state model for cylindrical geometry, but replacing the lateral dimension of the sample by the thickness, "R" $\rightarrow d = 10$ μm. For comparison, corresponding Bean model calculations are also plotted in Figure 4.4.1, as solid lines. The generalized critical state model best fits the data for $H_1 < H^*$, whereas the Bean model is best for $H_1 > H^*$.

Experimental harmonic power spectra of the pick-up signals at $H_1 = 10$, 21 and 40 Oe are shown as solid lines in Figures 4.4.2(a), 4.4.3(a) and 4.4.4(a), respectively. These spectra are all taken at zero dc magnetic field and 77 K. Unlike the Y-Ba-Cu-O thin-film spectra, the Bi-Sr-Ca-Cu-O single crystal spectra change significantly as a dc magnetic

field is added. In Figures 4.4.2(b), 4.4.3(b) and 4.4.4(b), data similar to Figures 4.4.2(a), 4.4.3(a) and 4.4.4(a), respectively, are taken, but at a dc magnetic field equal to 4 Oe. This means that the critical current is sensitive to small magnetic field. The dot-circles in Figures 4.4.2, 4.4.3 and 4.4.4 represent the calculated harmonic spectra according to the generalized critical state model, using $\beta = 1.3$, $H_0 = 3$ Oe and $H^* = 5.2$ Oe, while the dot-squares represent the Bean model calculation results. Surprisingly, although the generalized critical state model calculations give a more satisfactory fit for μ_1' and μ_1'' versus $\log_{10}(H_1)$ in Figure 4.4.1, the Bean model calculations fit the power spectra better in Figures 4.4.2, 4.4.3 and 4.4.4. This is not understood at present.

Odd harmonic power taken experimentally at 77 K as a function of $\log_{10}(H_1)$ are shown as circles in Figures 4.4.5 for $n = 3, 5, 7, 9$. These data are taken at zero dc field. Since J_c is dependent on H, application of dc magnetic field has a significant effect on the harmonics. For example, Figure 4.4.6 is a plot of $P(7f)$ versus $\log_{10}(H_1)$ at $H_{dc} = 10.2$ Oe. The solid and dashed lines are, respectively, calculations according to the Bean and generalized critical state model, using $\beta = 1.3$, $H_0 = 3$ Oe and $H^* = 5.2$ Oe. As for the Y-Ba-Cu-O thin-film in Section 4.3, model calculations give about the correct slopes and relative amplitudes but do not fit the details of the harmonic power versus ac field amplitude data satisfactorily. Presumably, the demagnetization effects of this geometry complicate the modeling problem significantly, especially in high harmonics because of their sensitivity to differences between the physical systems and the models.

For completeness, some harmonic power versus dc magnetic field data are shown in Figure 4.4.7. The representative harmonic numbers are 2, 8, 15 and 19; the ac driving frequency is 1 kHz, the amplitude is 30 Oe and the temperature is 77 K. The corresponding generalized critical state model calculations, with $\beta = 1.3$, $H_0 = 3$ Oe and $H^* = 5.2$ Oe, are shown in Figure 4.4.8. The high harmonics do not fit the data very well, probably because the higher the harmonic number, the more sensitive the

harmonic signal is to the fine details in the model. Like the $P(nf)$ versus $\log_{10}(H_1)$ data, and harmonic generation in disk-shaped superconductors in general, more work, both experimental and theoretical, will have to be done to understand the nonlinear electrodynamics of this sample.

4.5 Figure Captions and Figures of Chapter 4

Figure 4.1.1. Power spectra for powdered $YBa_2Cu_3O_{7-\delta}$ (sample no. C-15) at $T = 77$ K, driven by an ac field at frequency 7.7 kHz, of amplitude $H_1 = 2.3$ Oe. (Bottom) Dc field $H_{dc} = 0.0$ Oe. Odd harmonics $n = 3, 5, \ldots$ are generated. (Top) In a parallel dc field $H_{dc} = 1.0$ Oe, even harmonics $n = 2, 4, \ldots$ also appear, owing to symmetry breaking by the dc field. The vertical scale is 10 dBm per division.

Figure 4.1.2. Power spectrum for powdered $YBa_2Cu_3O_{7-\delta}$ (sample no. C-15) at $T = 77$ K, driven by an ac field at frequency 28 kHz, of amplitude $H_1 = 23$ Oe, in a dc field $H_{dc} \approx 1$ mOe. Odd harmonics up to $n = 41$ are clearly observed, as well as much weaker even harmonics. The broad background resonance at $f \approx 363$ kHz is that of the receiver coil itself.

Figure 4.1.3. Circles: $P(nf)$ versus harmonic number n from Figure 4.1.2; crossed circles: data from Figure 4.1.2 corrected for receiver coil resonance. Broken line: relative $P(nf)$ computed from the model Eqns. (2.1.4) and (2.1.9), $h_1 = 5.0$, adjusted to fit data at $n = 3$. Dotted line: $P(nf)$ computed for first-order model, Eqn. (2.1.13), with $h_1 = 5.0$, $L_0 = 0.35$, $\kappa = 0.3$, as in Figure 4.1.8.

Figure 4.1.4. (a) Second harmonic power $P(2f)$ versus dc field H_{dc} for $YBa_2Cu_3O_{7-\delta}$ (sample no. C-15) at $T = 77$ K, $H_1 = 2.3$ Oe, $f = 52.5$ kHz. (b) Expansion of the scan resolution by 2000×, showing narrow dip at $H_{dc} = 0$, of width $\triangle H_{dc} \sim 0.1$ mOe.

Figure 4.1.5. Relative harmonic power (10 dB per division) of $P(nf)$ versus H_{dc}, scanned at a uniform rate, for powdered $YBa_2Cu_3O_{7-\delta}$ (sample no. C-15), at $T = 77$ K, $H_1 = 23$ Oe, $f = 28$ kHz. (a) to (f) show data for selected representative harmonics n. Shown are two scans, the arrows denoting the direction of time increase. There is a small hysteresis with this property: if a leftward trace is reversed, it superposes exactly

on the rightward trace. The sharp dips are interpreted as evidence for a pseudo flux quantization of an ensemble of supercurrent loops in the granular sample.

Figure 4.1.6. Relative harmonic power (10 dB per division) of $P(nf)$ versus h_{dc}, computed from zero-order model Eqn. (2.1.4), using $h_1 = 5.0$ and $F(A)$ from Eqn. (2.1.9). (a) to (f) show the same harmonic numbers as for the data, Figure 4.1.5.

Figure 4.1.7. (a) Average spacing $\triangle H_{dc}$ between dips from the data of Figure 4.1.5, versus harmonic number n. (b) Average spacing $\triangle h_{dc}$ between dips versus harmonic number n, computed from Eqns. (2.1.4) and (2.1.6) for various values of the standard deviation σ of the assumed Gaussian distribution.

Figure 4.1.8. Relative harmonic power (10 dB per division) of $P(nf)$ versus h_{dc}, computed from the first-order model, Eqn. (2.1.13), using parameters $h_{dc} = 5$, $L_0 = 0.35$, $\kappa = 0.30$, and $F(A)$ from Eqn. (2.1.9).

Figure 4.1.9. $|Q_n(h_1, A)| \equiv |A\, J_n(Ah_1)\, F(A)|$ versus A, in dimensionless units; $F(A)$ from Eqn. (2.1.9). (a) $n = 10$, $h_1 = 5.0$; (b) $n = 5$, $h_1 = 5.0$; (c) $n = 5$, $h_1 = 10.0$.

Figure 4.1.10. Relative harmonic power (10 dB/div) $P(16f)$ versus H_{dc} for powdered Y-Ba-Cu-O sample, no. C-15, at $T = 77$ K, $f = 28$ kHz, for a series of values of the ac field H_1; the arrows denote the direction of increasing time in the scan. If leftward trace (c) is reversed and plotted (d), it superposes exactly on the rightward trace (e).

Figure 4.1.11. Circles: Measured values of the dc field H_{dc} for a sharp dip in $P(15f)$ as a function of the ac field H_1; taken for sample no. C-15 at 77 K. Diamonds denote less well defined dips. Solid lines: h_1 versus h_{dc}, where $h_{dc} \propto H_{dc}$ is the computed position of a sharp dip as a function of $h_1 \propto H_1$. Computation is made from Eqns. (2.1.4) and (2.1.9).

Figure 4.1.12. Relative harmonic power (10 dB/div) for $P(2f)$ versus H_{dc} for

sample no. C-15, powdered Y-Ba-Cu-O, at $T = 77$ K, showing well-defined transition, with structure, as H_1 field is reduced. Similar behavior is observed for $n = 4$.

Figure 4.1.13. Relative harmonic power (10 dB/div) for $P(2f)$ versus h_{dc}, computed from the "loop" model, Eqns. (2.1.12) and (2.1.9). An abrupt transition to structure is observed as h_1 is reduced; cf. Figure 4.1.12.

Figure 4.2.1. (a) Measured harmonic power $P(nf)$ versus H_{dc} for $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N). $T = 77$ K, $H_1 = 13.5$ Oe, and $f = 10.5$ kHz. Vertical scale division equals to 10 dB. (b) Modified critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. (c) Same as (a) except $H_1 = 4.5$ Oe. (d) Solid lines: Same as (b), except $H_1 = 4.5$ Oe; for $n = 10$ the dashed line is that predicted for $\beta = 1.0$, and does not fit the data.

Figure 4.2.2. (a), (b) Open circles: measured (Gaussian units) $\tilde{\mu}_{total}$ versus H_1 for $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 0$, $T = 77$ K, $f = 85$ Hz. Solid line: modified critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe, for intergranular component $\tilde{\mu}$. Dashed line: model calculation for intragranular component $\tilde{\mu}_g$ (reduced by a factor of 3 for μ''_g) using parameters $\beta_g = 1.0$, $H_{0g} = 5$ Oe, $H^*_{intra} = 250$ Oe, $\overline{R_g} = 10$ μm. (c) Measured μ''_{total} versus H_1 for powdered $YBa_2Cu_3O_{7-\delta}$ (sample no. C-48B), $H_{dc} = 0$, $T = 77$ K, $f = 85$ Hz. The intergranular component is absent; cf. (a). (d) Same as (a) but with $H_{dc} = 10.2$ Oe, which splits the intergranular component.

Figure 4.2.3. (a) Circles: Third harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 has been amplified $100\times$ (+40 dB) by the PARC 113 preamplifier before going to the spectrum analyzer. Solid line: generalized critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. Dot-dashed line: generalized critical model predictions with $\beta_g = 1.0$,

$H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. Dashed line: generalized critical model predictions with $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. (b) Third harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ powder (sample no. C-48B), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 has been amplified 10× (+20 dB) by the PARC 113 preamplifier before going to the spectrum analyzer. (c) Circles: Fifth harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Solid line: generalized critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. Dot-dashed line: generalized critical state model predictions with $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. Dashed line: generalized critical state model predictions with $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. (d) Fifth harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ powder (sample no. C-48B), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification.

Figure 4.2.4. (a) Circles: Seventh harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Solid line: generalized critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. Dot-dashed line: generalized critical state model predictions with $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. Dashed line: generalized critical state model predictions with $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. (b) Corresponding experimental measurements and model calculations for the ninth harmonic power. Specifications are the same as (a).

Figure 4.2.5. Open circles: Experimentally measurements (Gaussian units) of (a) μ_1' and (b) μ_1'' versus $\log_{10}(H_1)$ for $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N),

H_{dc} = 51.4 Oe, T = 77 K, f = 85 Hz. Solid line: modified critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe, for intergranular component $\tilde{\mu}$. Dashed line: model calculation for intragranular component $\tilde{\mu}_g$ (reduced by a factor of 3 for μ_g'') using parameters $\beta_g = 1.0$, $H_{0g} = 5$ Oe, $H^*_{intra} = 250$ Oe, $\overline{R_g} = 10$ μm.

Figure 4.2.6. Same as Figure 4.2.5, except that $H_{dc} = 60.3$ Oe.

Figure 4.2.7. Same as Figure 4.2.5, except that $H_{dc} = 70.2$ Oe.

Figure 4.2.8. Same as Figure 4.2.5, except that $H_{dc} = 90.4$ Oe.

Figure 4.2.9. (a) Circles: Seventh harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C46N), H_{dc} = 10.1 Oe, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Solid line: generalized critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. Dot-dashed line: generalized critical state model predictions with $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. Dashed line: generalized critical state model predictions with $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. (b) Circles: Fifth harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 30.2$ Oe, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Solid line: generalized critical state model predictions with $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe. Dot-dashed line: generalized critical state model predictions with $\beta_g = 1.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. Dashed line: generalized critical state model predictions with $\beta_g = 2.0$, $H_{0g} = 5$ Oe and $H^*_{intra} = 250$ Oe. (c) Same as (b), except that $H_{dc} = 60.4$ Oe. (d) Same as (b), except that $H_{dc} = 91.5$ Oe.

Figure 4.2.10. Harmonic spectra generated by $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N) taken at 78.5 K and $H_{dc} = 0$. The ac magnetic field amplitudes H_1 are (a) 5.0 Oe, (b) 10.0 Oe, (c) 20.0 Oe and (d) 40.0 Oe. The squares are the harmonic

spectra predicted by the generalized critical state model calculations for the intergranular component only: $\beta = 1.8$, $H_0 = 3$ Oe and $H^* = 15$ Oe.

Figure 4.2.11. Experimental measurements (Gaussian units) of (a) μ_1' and (b) μ_1'' versus $\log_{10}(H_1)$ for $YBa_2Cu_3O_{7-\delta}$ sintered rod (sample no. C-46N), $H_{dc} = 0$, $T = 78.8$ K, $f = 85$ Hz. The experimental value of the effective permeability of the macroscopic polycrystalline medium μ_{eff} is indicated. H^*_{inter} and H^*_{intra}, respectively, are the external field at which the flux penetrates to the centers of the sintered rod (intergranular medium) and the individual superconducting grains in the ceramic.

Figure 4.2.12. Same as Figure 4.2.11, except that $T = 83.7$ K.

Figure 4.2.13. Same as Figure 4.2.11, except that $T = 86.0$ K.

Figure 4.2.14. Same as Figure 4.2.11, except that $T = 87.5$ K.

Figure 4.2.15. Same as Figure 4.2.11, except that $T = 89.5$ K.

Figure 4.2.16. The measured effective permeability, μ_{eff}, of the intergranular medium of the sintered $YBa_2Cu_3O_{7-\delta}$ cylinder (sample no. C-46N), is plotted as a function of temperature for the data of Figures 4.2.11 to 4.2.15, plus 13 other temperature values in the range 77 to 91.5 K. The dashed line is the calculation of μ_{eff} according to Eqns. (4.2.2) and (4.2.3), with $f_n = 0.144$, $\lambda_g(T = 0) = 1400$ Å, $T_c = 91.2$ K, and $\overline{R_g} = 10$ μm.

Figure 4.2.17. The inter- and intragranular penetration fields, H^*_{inter} (multiplied by 10) and H^*_{intra}, are plotted versus temperature. The solid lines are linear regression fits to the measured data, and yield the intrinsic critical temperature of 91.2 K and the intergranular phase-locking temperature of 86.6 K. The data set is the same as for Figure 4.2.16.

Figure 4.2.18. The zero-field inter- and intragranlar critical current densities, $J_{c,J}(H = 0)$ and $J_{c,g}(H = 0)$, as derived from the H^*_{inter} and H^*_{intra} data in Figure 4.2.17 using the generalized critical state model result, Eqn. (4.2.4), are plotted

versus temperature. The solid lines are only guides to the eyes. The specific parameters used in the derivations are: $\beta_J = 1.8$, $H_{0,J} = 3$ Oe and $\beta_g = 1.0$, $H_{0,g} = 5$ Oe.

Figure 4.3.1. Open circles: Experimental measurements (Gaussian units) of (a) μ_1' and (b) μ_1'' versus $\log_{10}(H_1)$ for $YBa_2Cu_3O_{7-\delta}$ thin-film (sample no. C-51), $H_{dc} = 0$, $T = 77$ K, $f = 85$ Hz. Solid line: Bean model predictions, Eqn. (2.2.6), with $H^* = 95$ Oe and "R" $= 0.5$ μm, the film thickness.

Figure 4.3.2. Harmonic spectra generated by $YBa_2Cu_3O_{7-\delta}$ thin-film (sample no. C-51) taken at 77 K and $H_{dc} = 0$. The ac magnetic field amplitudes H_1 are (a) 5.0 Oe, (b) 12.0 Oe, (c) 50.0 Oe and (d) 100.0 Oe. The squares are the harmonic spectra predicted by the Bean critical state model calculations with $H^* = 95$ Oe.

Figure 4.3.3. (a) Circles: Third harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ thin-film (sample no. C-51), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 has been amplified by 50× (+34 dB) by a PARC 113 preamplifier before going to the spectrum analyzer. Solid line: Bean critical state model predictions with $H^* = 95$ Oe. (b) Circles: Fifth harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $YBa_2Cu_3O_{7-\delta}$ thin-film (sample no. C-51), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Solid line: Bean critical state model predictions with $H^* = 95$ Oe. (c) and (d) Circles and solid lines: corresponding experimental measurements and Bean model calculations for the seventh and the ninth harmonic power. Other details are the same as (b). The diamonds in (c) are experimental data taken at $H_{dc} = 100$ Oe; there is little change from the data taken at $H_{dc} = 0$ (circles).

Figure 4.4.1. Open circles: Experimental measurements (Gaussian units) of (a) μ_1' and (b) μ_1'' versus $\log_{10}(H_1)$ for $Bi_2Sr_2CaCu_2O_8$ single crystal (sample no. C-50), $H_{dc} = 0$, $T = 77$ K, $f = 85$ Hz. Dashed line: generalized critical state model

calculations, with $\beta = 1.3$, $H_0 = 3$ Oe, and $H^* = 5.2$ Oe. Solid line: Bean model predictions, Eqn. (2.2.6), with $H^* = 5.2$ Oe.

Figure 4.4.2. Harmonic spectra generated by $Bi_2Sr_2CaCu_2O_8$ single crystal (sample no. C-50) taken at $T = 77$ K, $f = 1.0$ kHz, and ac field amplitude $H_1 = 10.0$ Oe. The dc magnetic fields H_{dc} are (a) 0.0 Oe, (b) 4.0 Oe. The circles are the harmonic spectra predicted by the generalized critical state model calculations with $\beta = 1.3$, $H_0 = 3$ Oe, and $H^* = 5.2$ Oe. The squares are the Bean critical state model calculations with $H^* = 5.2$ Oe. Note that since the Bean model, J_c is independent of the magnetic field, no even harmonics are generated even when $H_{dc} \neq 0$.

Figure 4.4.3. Same as Figure 4.4.2, except that $H_1 = 21.0$ Oe.

Figure 4.4.4. Same as Figure 4.4.2, except that $H_1 = 40.0$ Oe.

Figure 4.4.5. (a) Circles: Third harmonic power (10 dB/div) versus $\log_{10}(H_1)$ experimentally measured on $Bi_2Sr_2CaCu_2O_8$ single crystal (sample no. C-50), $H_{dc} = 0$, $T = 77$ K, $f = 1.0$ kHz. The balanced signal from Coils #5 and #6 goes to the spectrum analyzer without preamplification. Dashed line: generalized critical state model calculations, with $\beta = 1.3$, $H_0 = 3$ Oe, and $H^* = 5.2$ Oe. Solid line: Bean critical state model predictions with $H^* = 5.2$ Oe. (b) Corresponding measurements and calculations for the fifth harmonic power. Specifics the same as (a). (c) Corresponding measurements and calculations for the seventh harmonic power. Specifics the same as (a). (d) Corresponding measurements and calculations for the ninth harmonic power. Specifics the same as (a).

Figure 4.4.6. Same as Figure 4.4.5(c), except that $H_{dc} = 10.2$ Oe.

Figure 4.4.7. Relative harmonic power (10 dB per division) of $P(nf)$ versus H_{dc}, scanned at a uniform rate, for $Bi_2Sr_2CaCu_2O_8$ single crystal (sample no. C-50), at $T = 77$ K, $H_1 = 30$ Oe, $f = 1.0$ kHz. (a) to (d) show data for selected representative harmonics: $n = 2$, 8, 15, 19. Shown are two scans, the arrows denoting the direction

of time increase. There is a small hysteresis with this property: if a leftward trace is reversed, it superposes exactly on the rightward trace.

Figure 4.4.8. Generalized critical state model calculations corresponding to the harmonic numbers and ac magnetic field amplitude used in Figure 4.4.7. The parameters used in the calculations are: $\beta = 1.3$, $H_0 = 3$ Oe, and $H^* = 5.2$ Oe.

YBCO powder

H_1 = 2.3 Oe T = 77 K f = 7.7 kHz

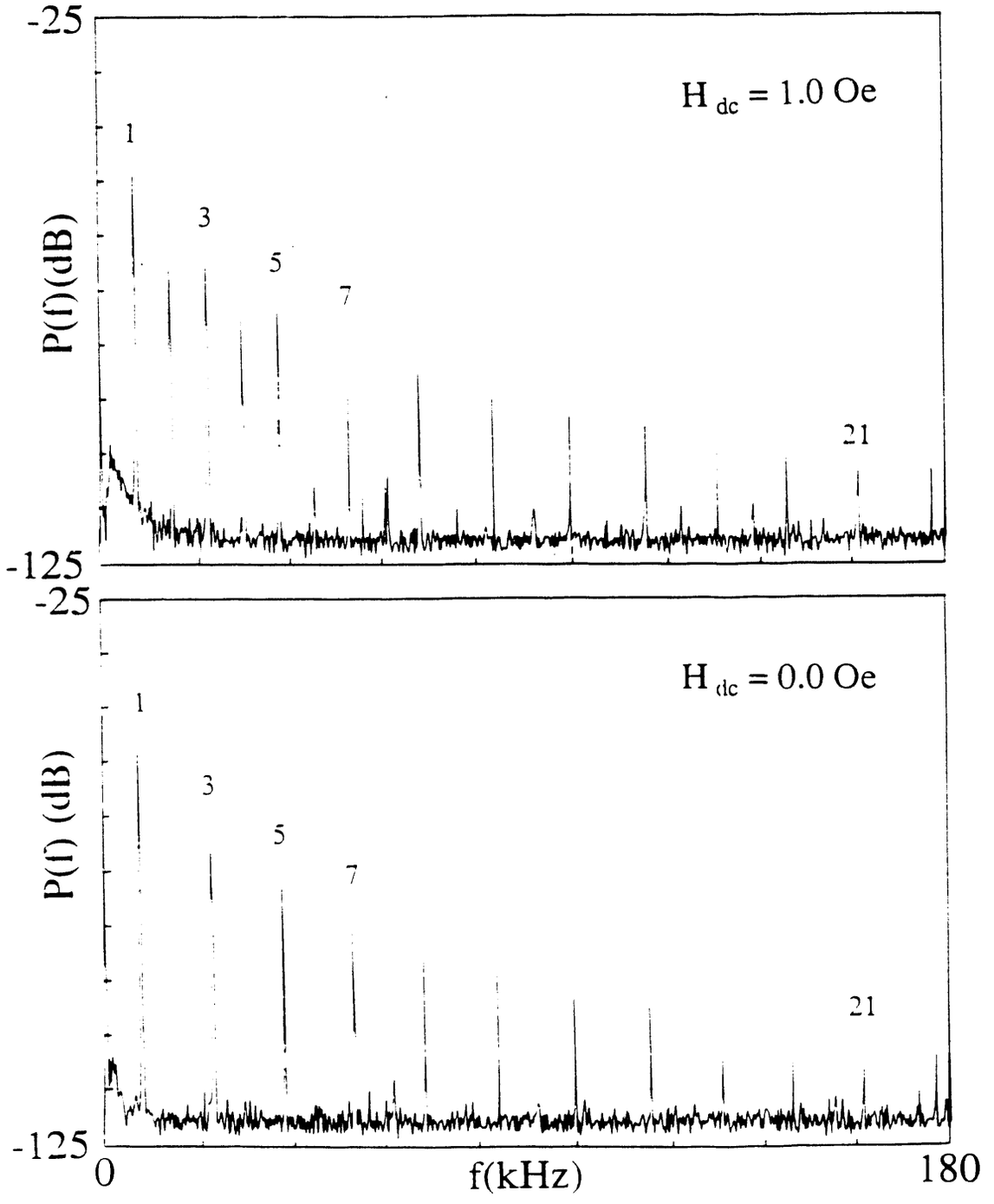


Figure 4.1.1

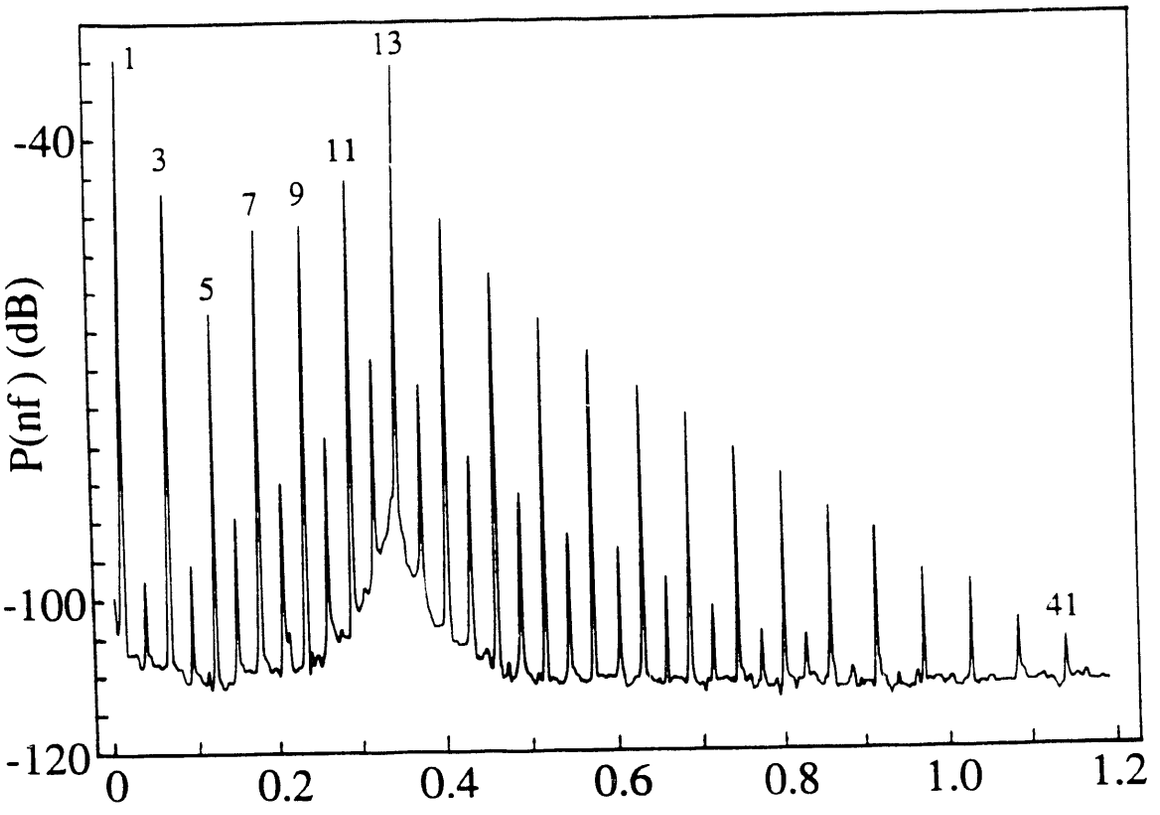


Figure 4.1.2

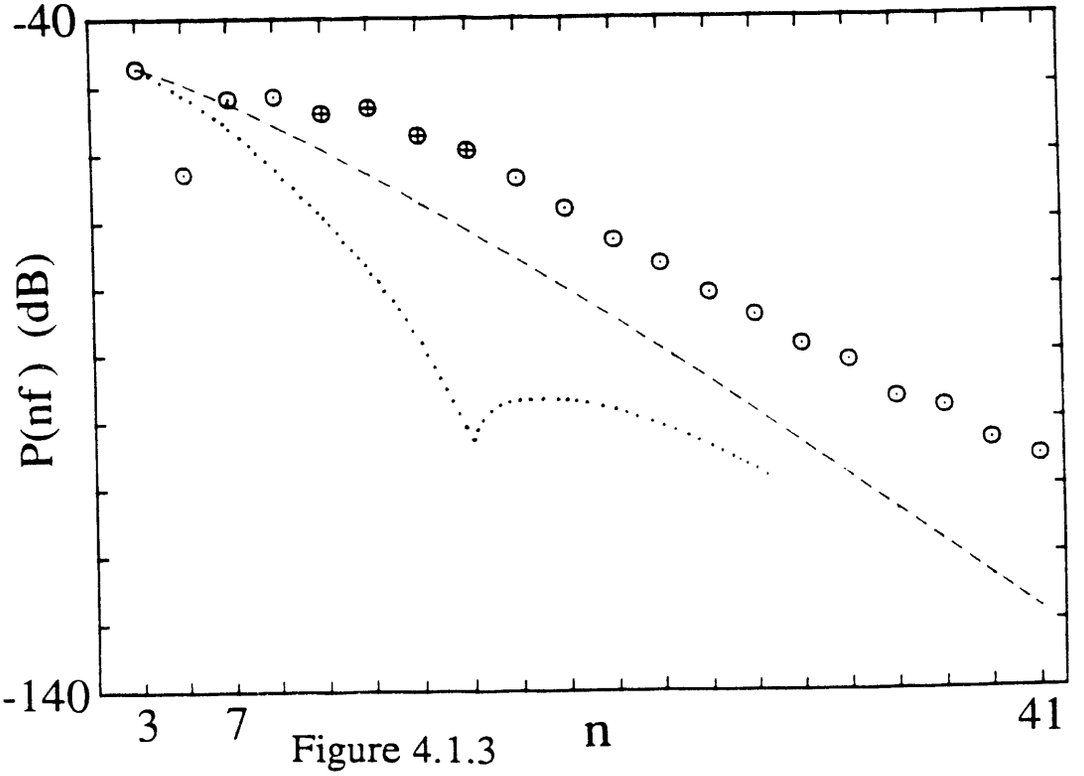
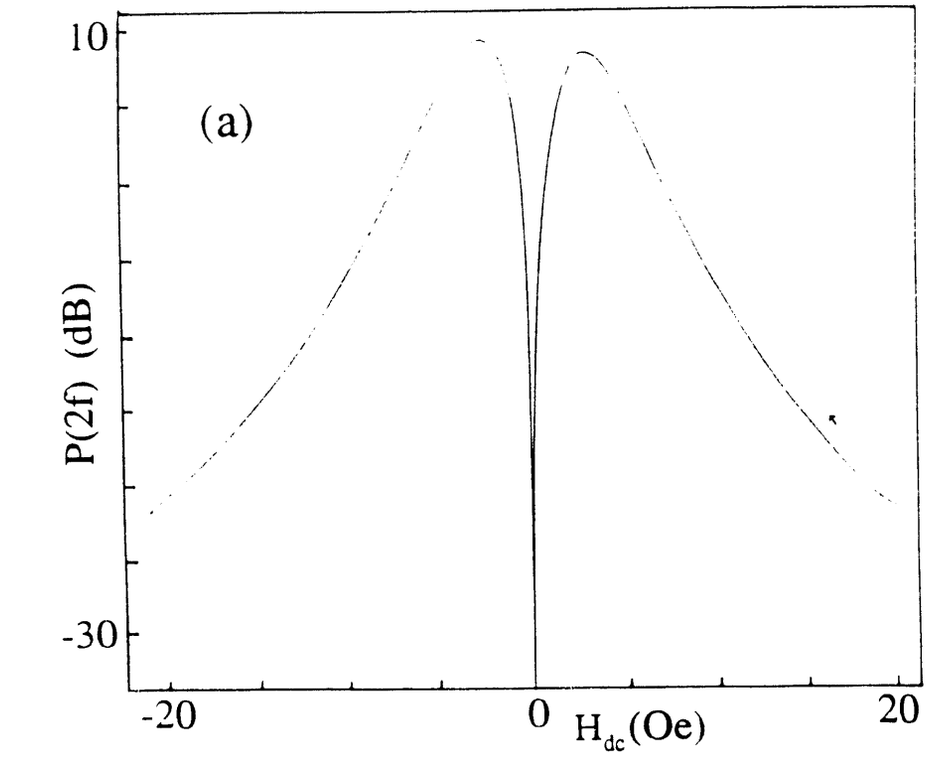


Figure 4.1.3



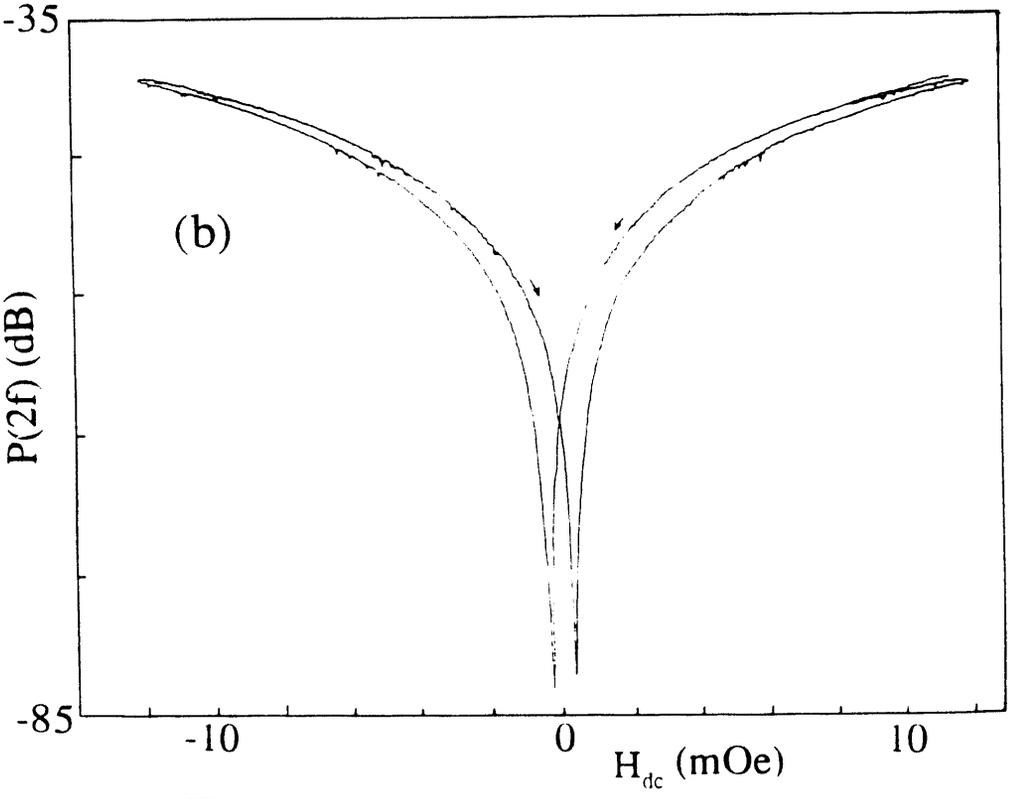


Figure 4.1.4

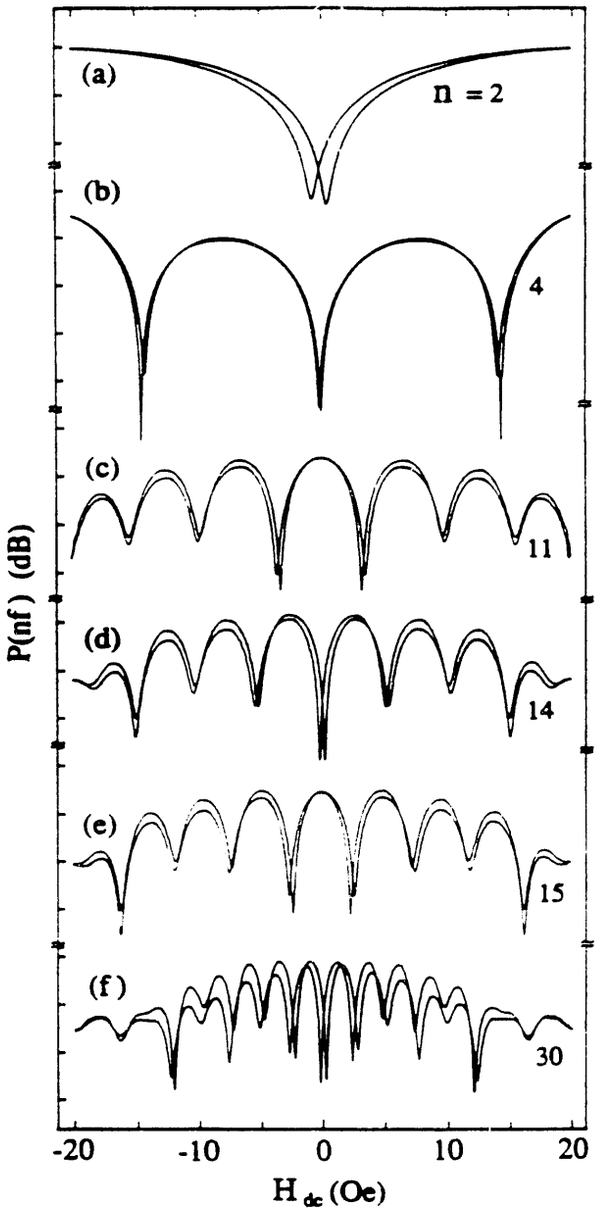


Figure 4.1.5

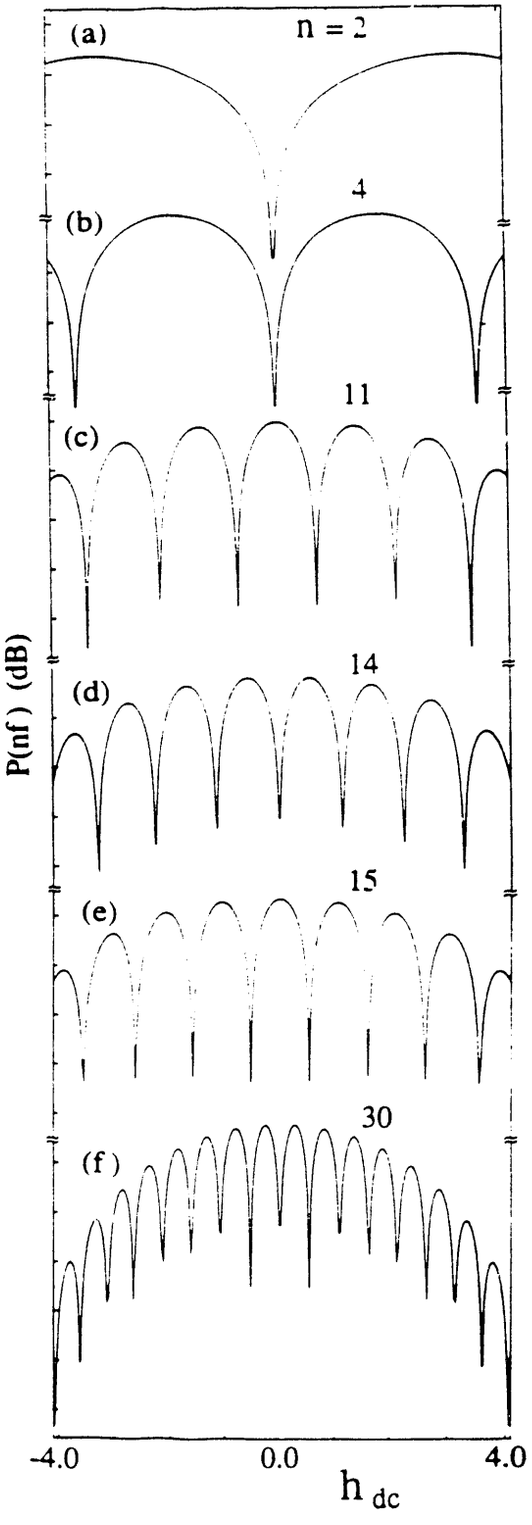
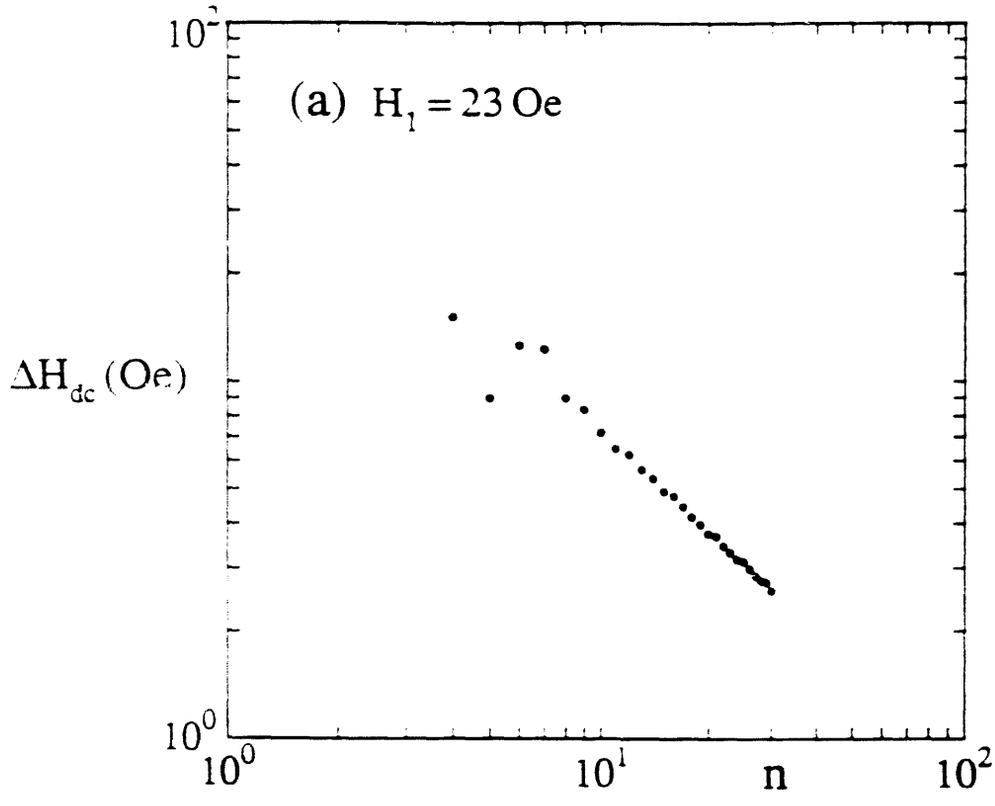


Figure 4.1.6


(a) $H_1 = 23$ Oe
ΔH_{dc} (Oe)
10^2
10^0
10^0
10^1
10^2
n

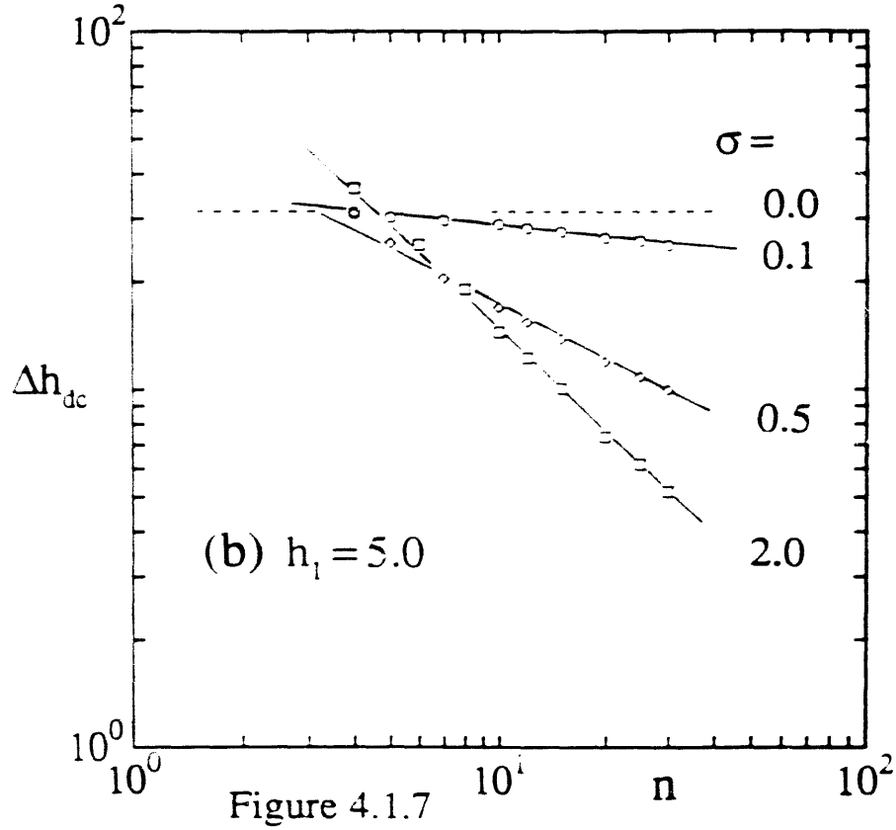

$\sigma =$
0.0
0.1
0.5
2.0
(b) $h_1 = 5.0$
Δh_{dc}
10^2
10^0
10^0
10^1
10^2
n

Figure 4.1.7

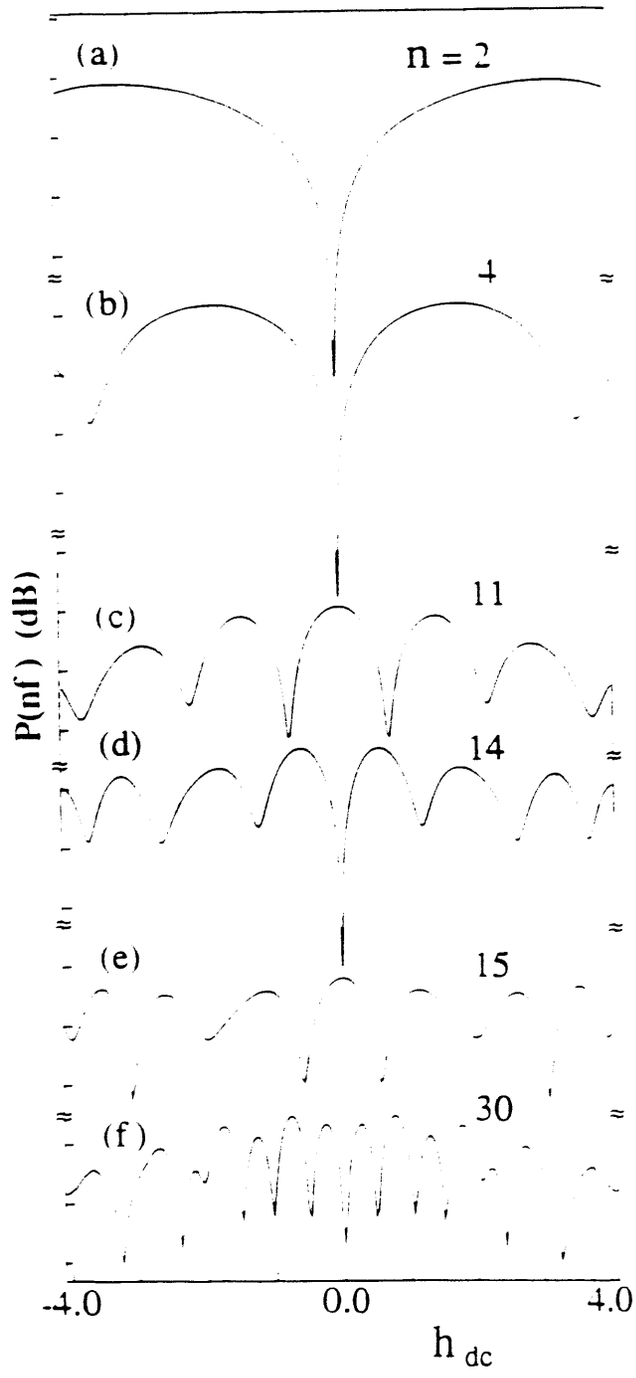


Figure 4.1.8

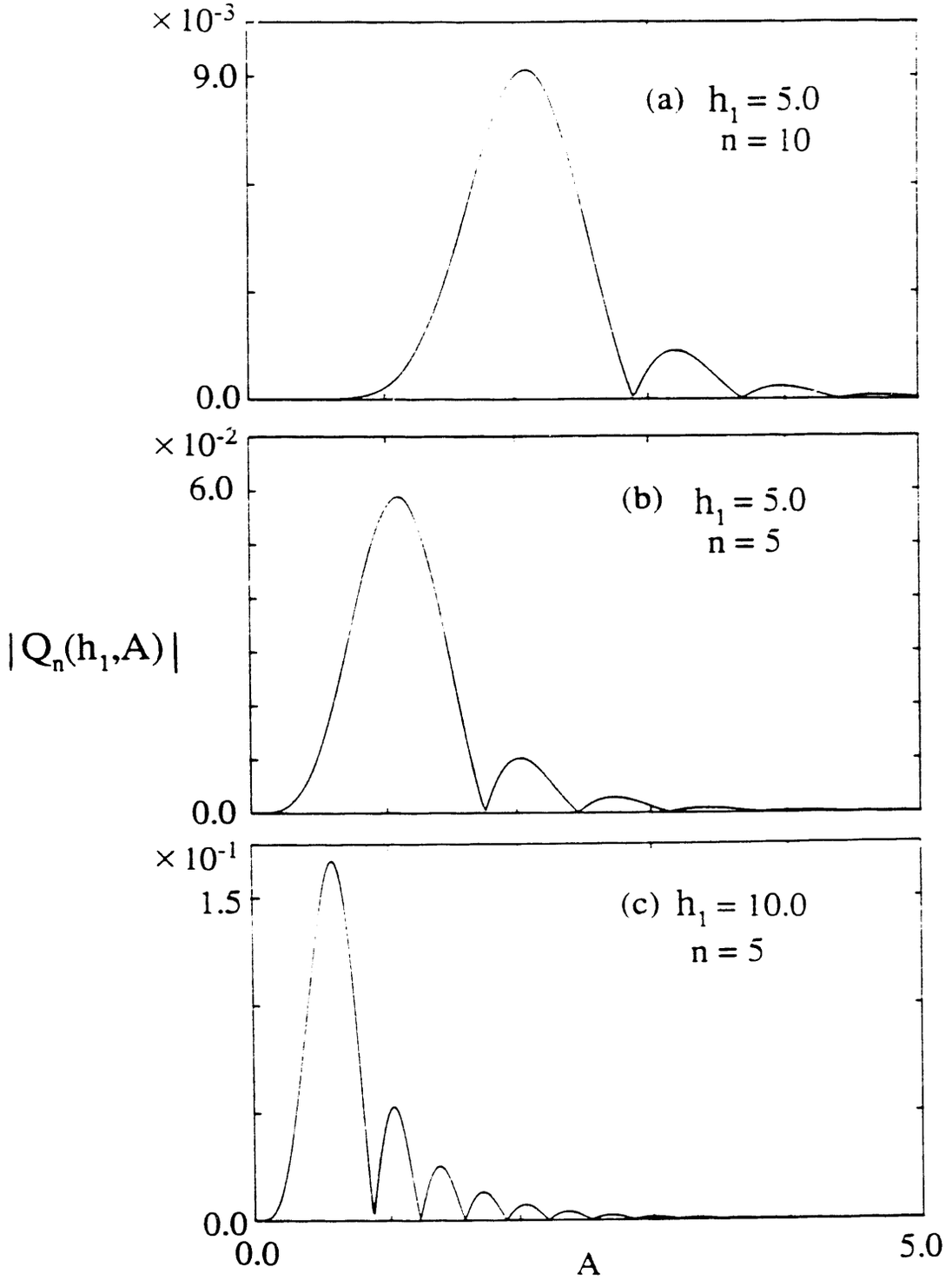

× 10-3
9.0
0.0
(a) h1 = 5.0
n = 10
× 10-2
6.0
0.0
(b) h1 = 5.0
n = 5
|Qn(h1,A)|
× 10-1
1.5
0.0
(c) h1 = 10.0
n = 5
0.0
5.0
A

Figure 4.1.9

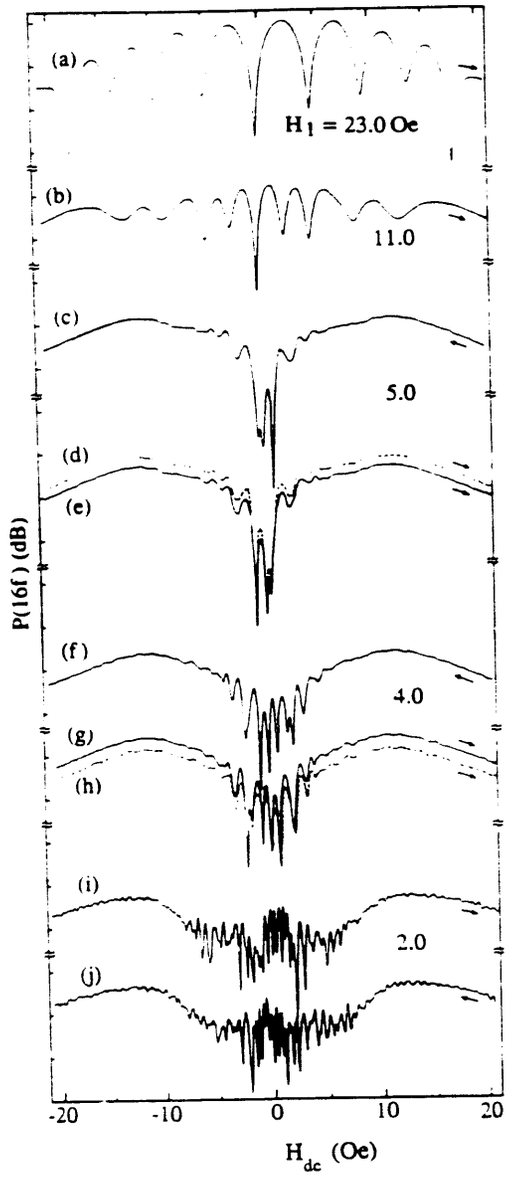


Figure 4.1.10

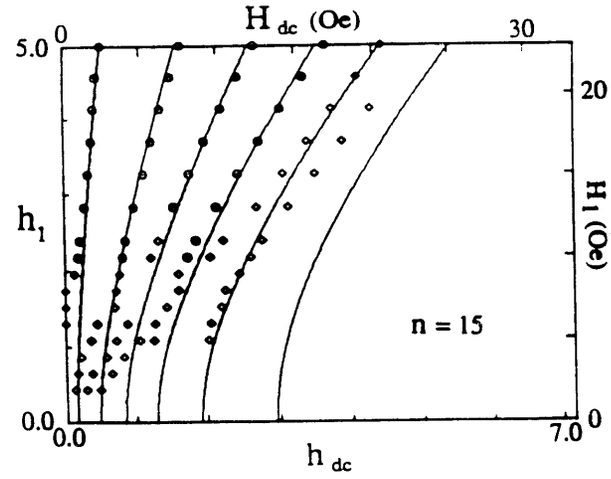


Figure 4.1.11

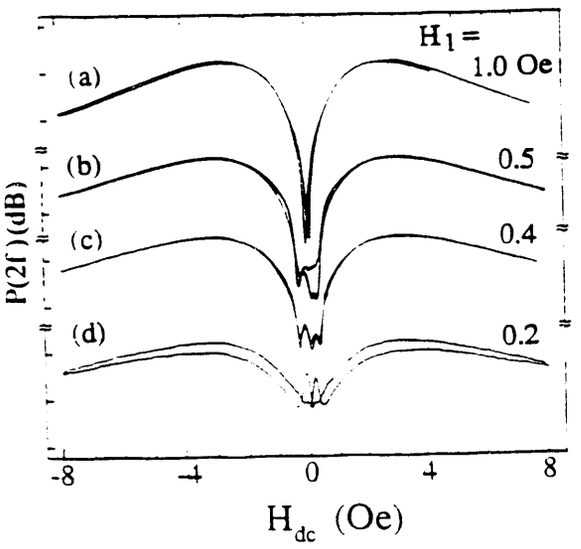


Figure 4.1.12

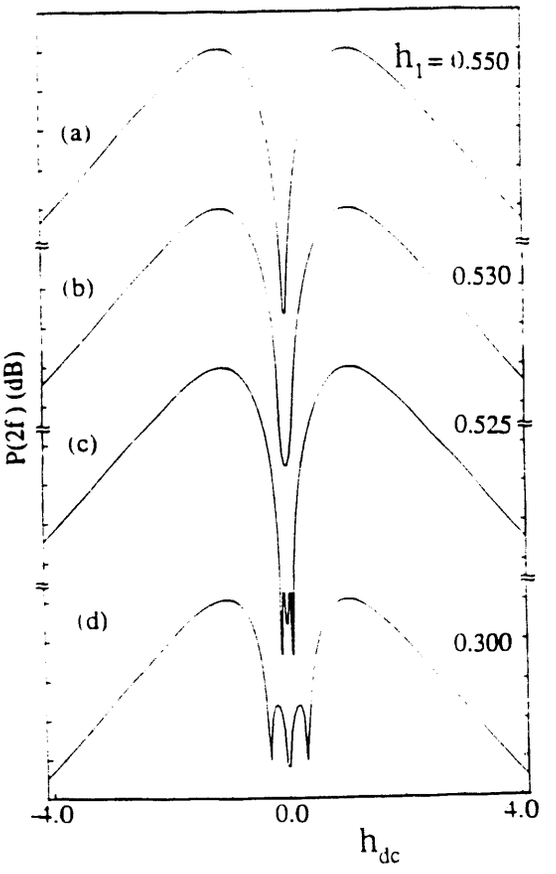


Figure 4.1.13

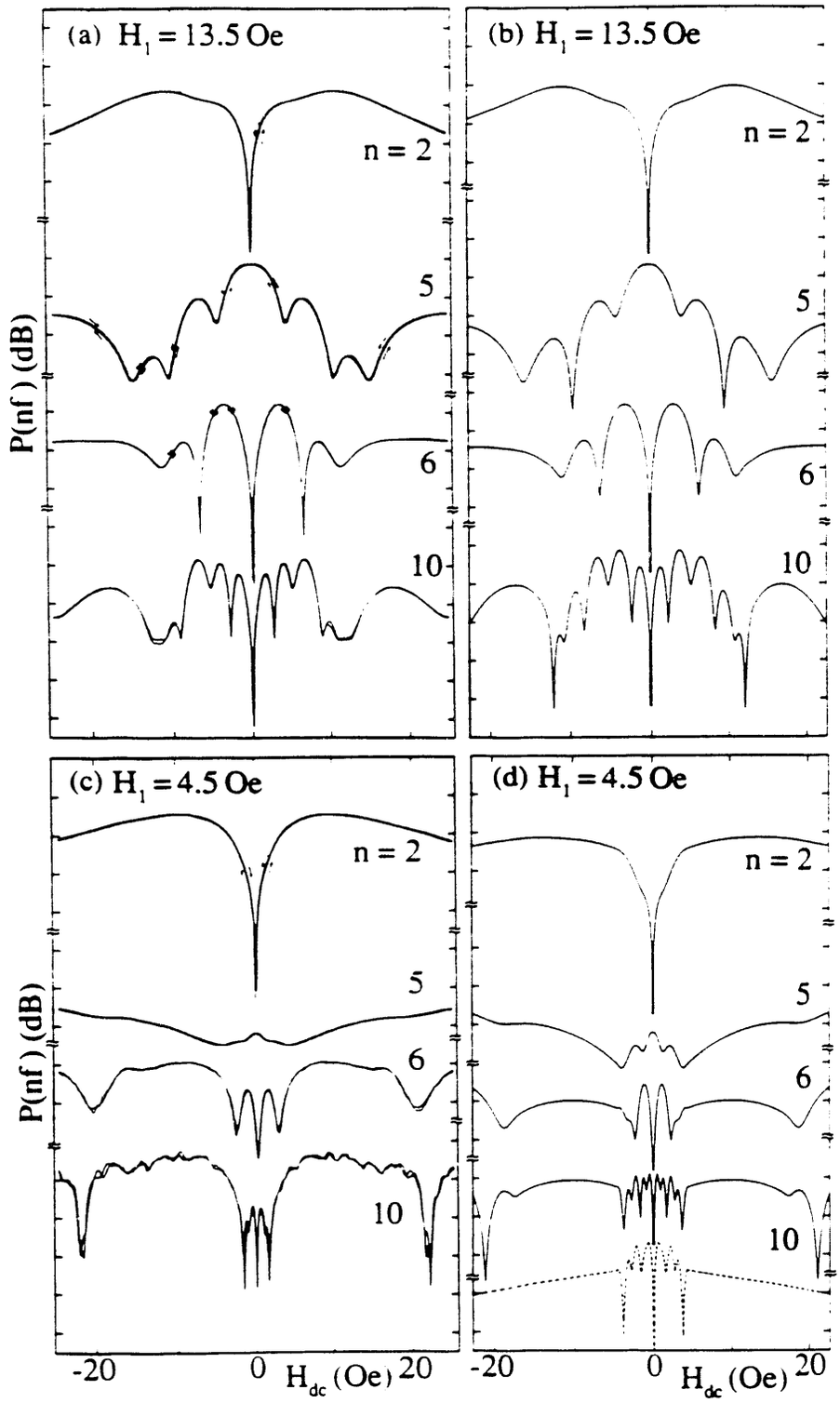

(a) $H_1 = 13.5$ Oe
(b) $H_1 = 13.5$ Oe
(c) $H_1 = 4.5$ Oe
(d) $H_1 = 4.5$ Oe
n = 2
5
6
10
P(nf) (dB)
-20
0
20
H_{dc} (Oe)

Figure 4.2.1

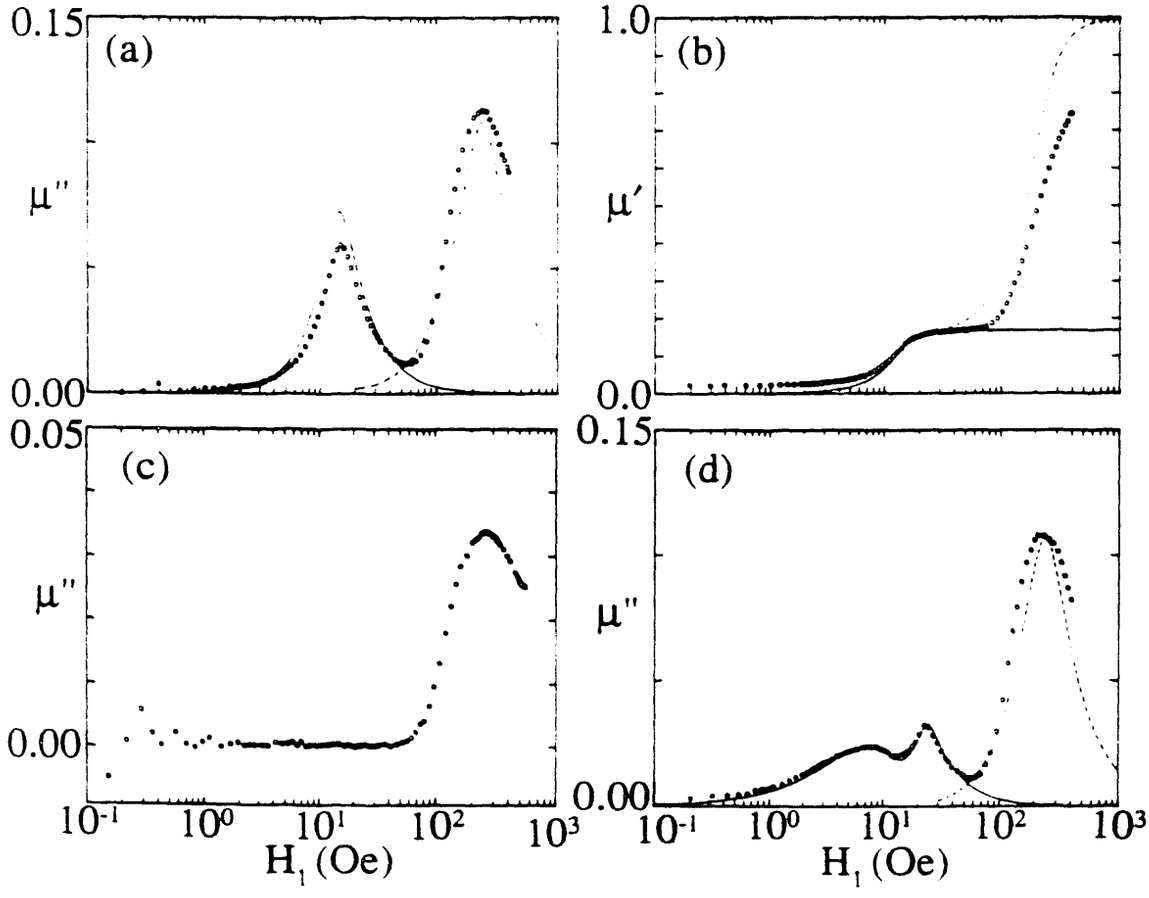

0.15
(a)
μ''
0.00
1.0
(b)
μ'
0.0
0.05
(c)
μ''
0.00
0.15
(d)
μ''
0.00
10-1
100
101
102
103
H1 (Oe)
10-1
100
101
102
103
H1 (Oe)

Figure 4.2.2

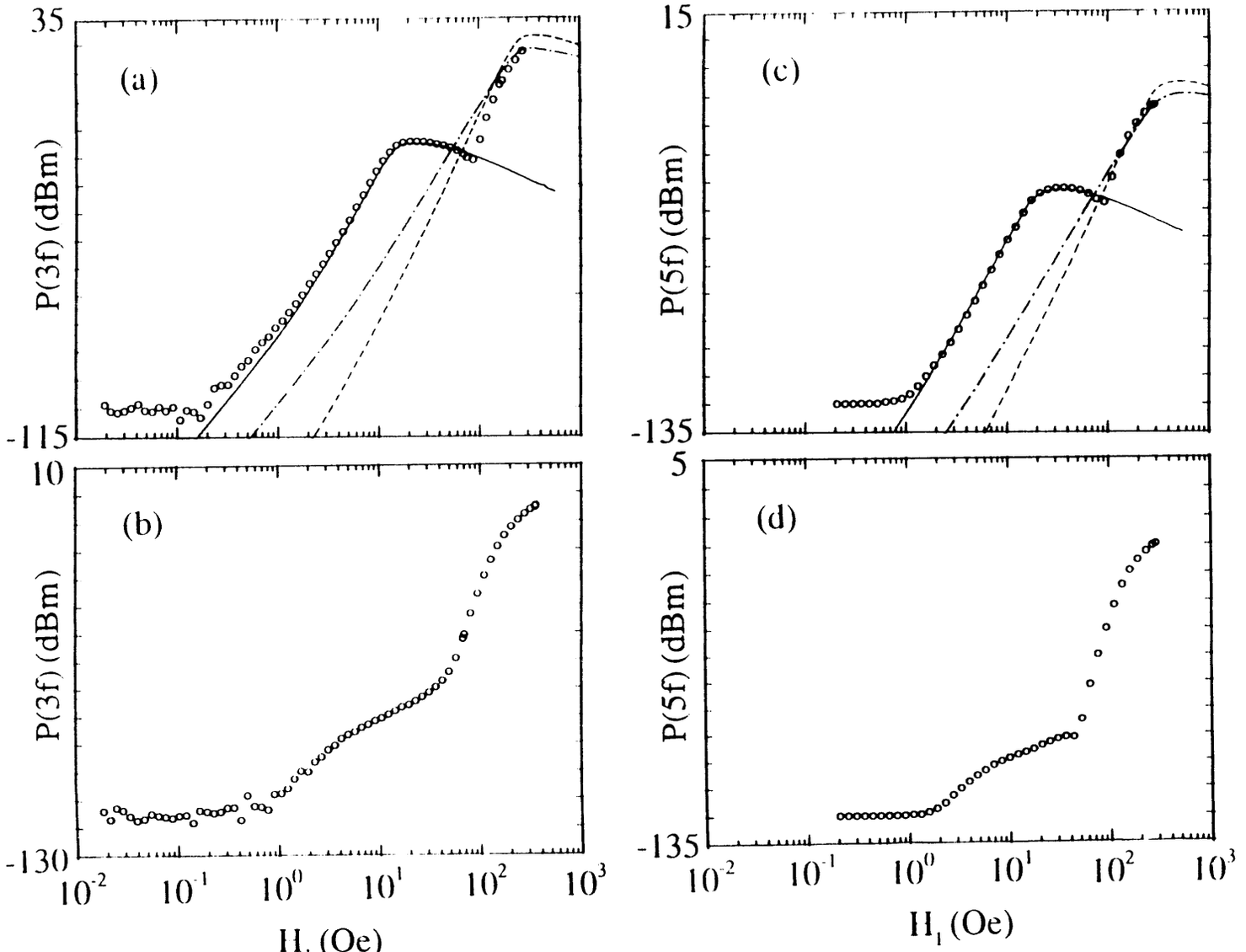

35
(a)
P(3f) (dBm)
-115
15
(c)
P(5f) (dBm)
-135
10
(b)
P(3f) (dBm)
-130
5
(d)
P(5f) (dBm)
-135
10^{-2} 10^{-1} 10^{0} 10^{1} 10^{2} 10^{3}
H_1 (Oe)
10^{-2} 10^{-1} 10^{0} 10^{1} 10^{2} 10^{3}
H_1 (Oe)

Figure 4.2.3

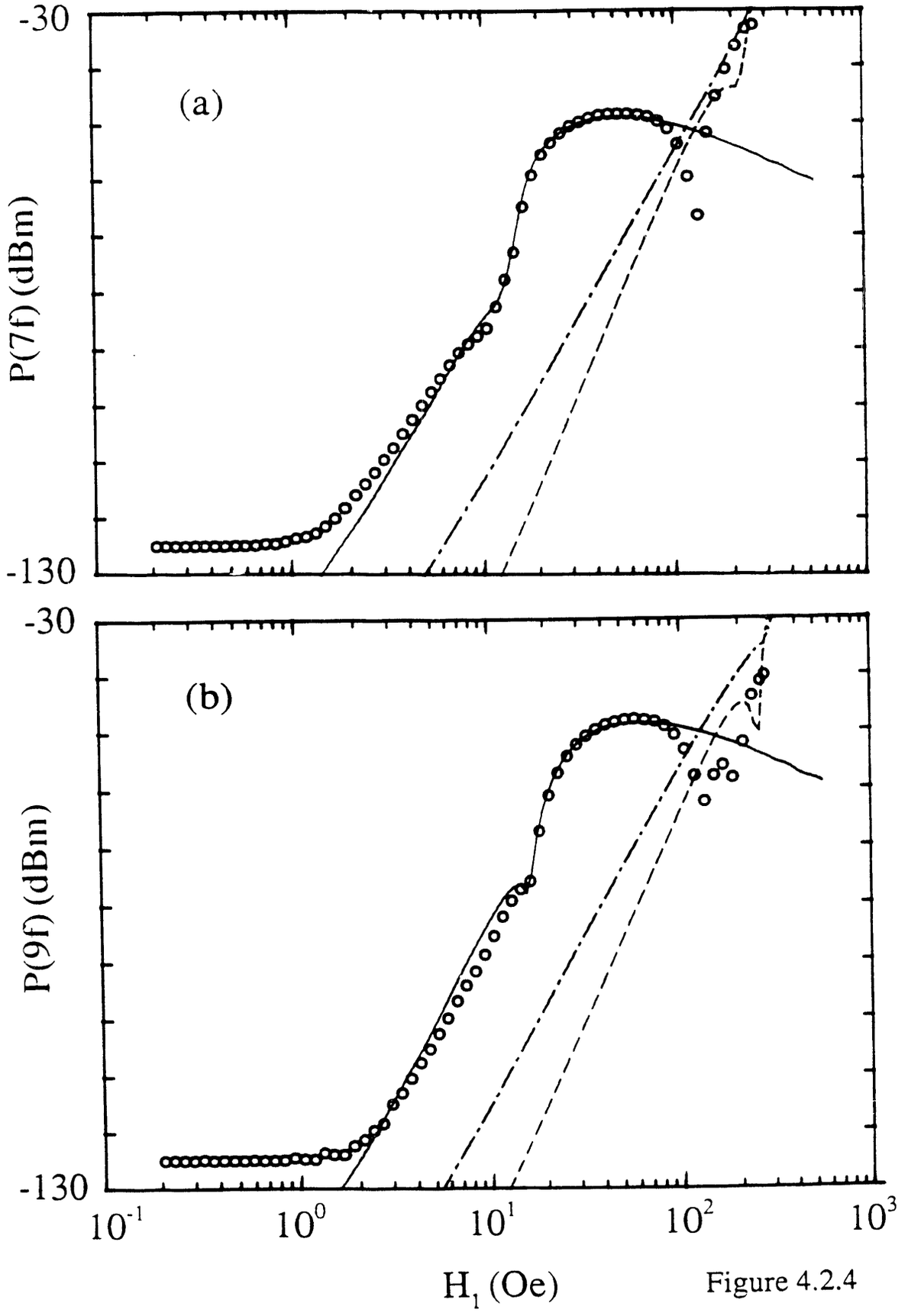


Figure 4.2.4

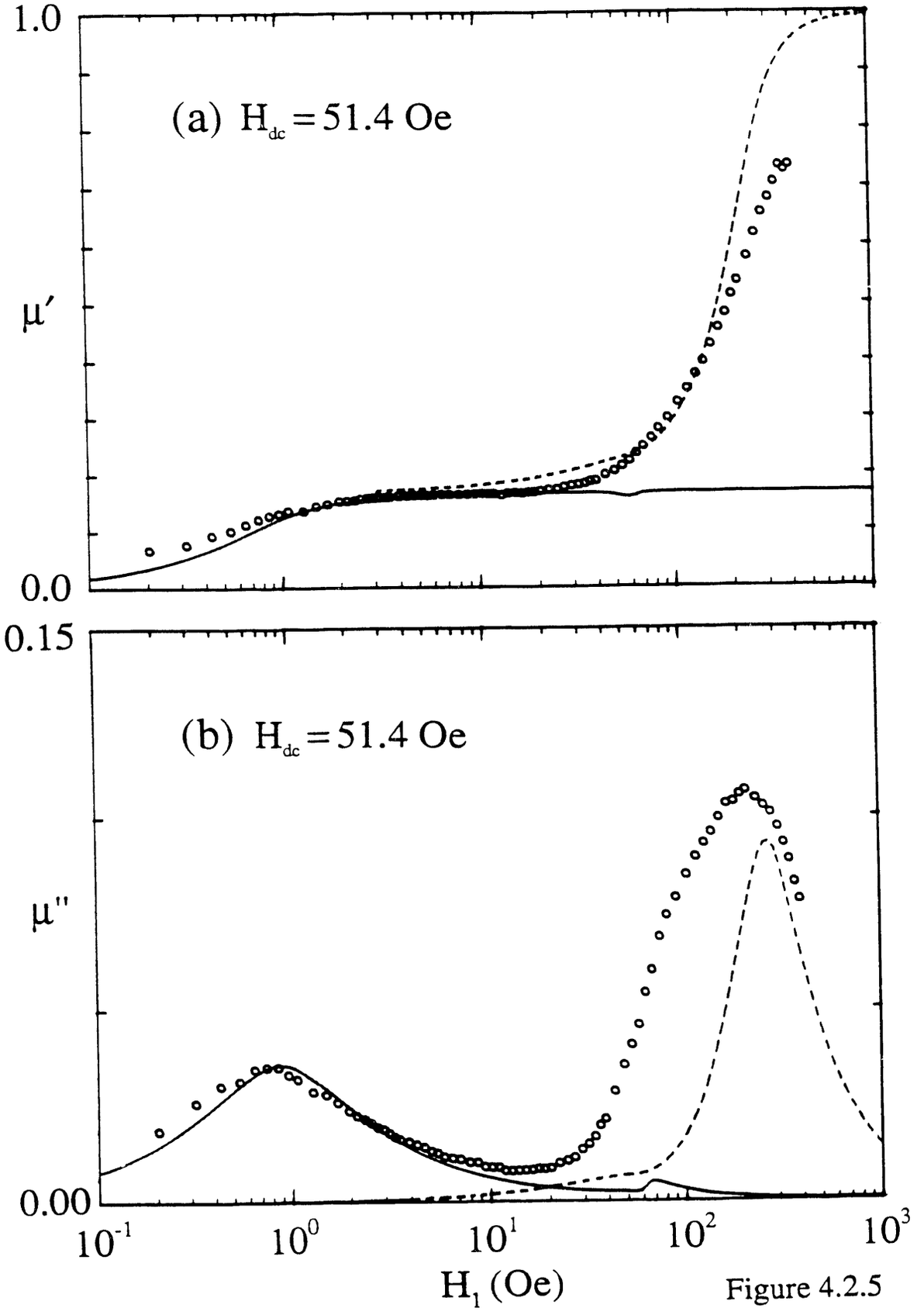


Figure 4.2.5

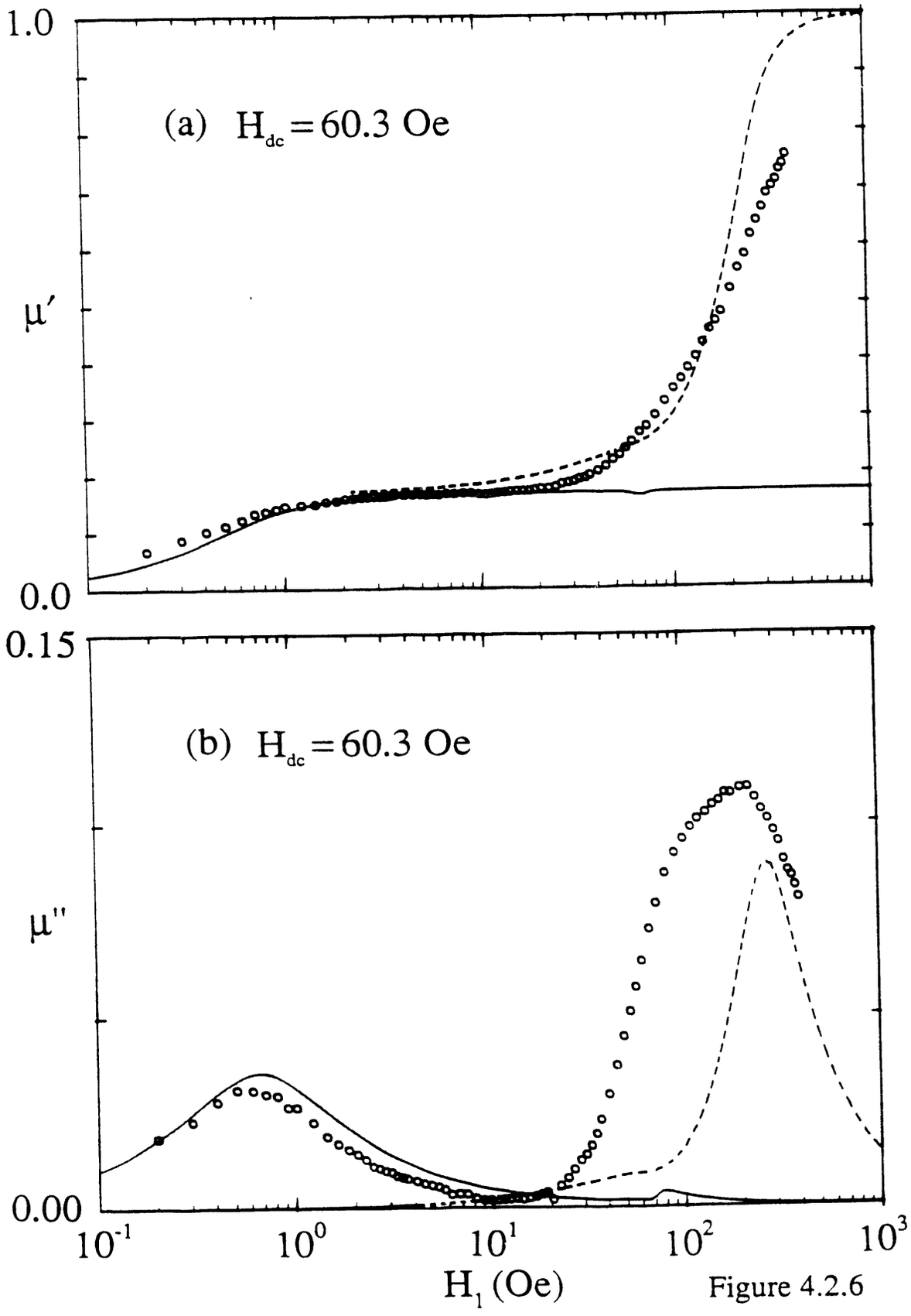


Figure 4.2.6

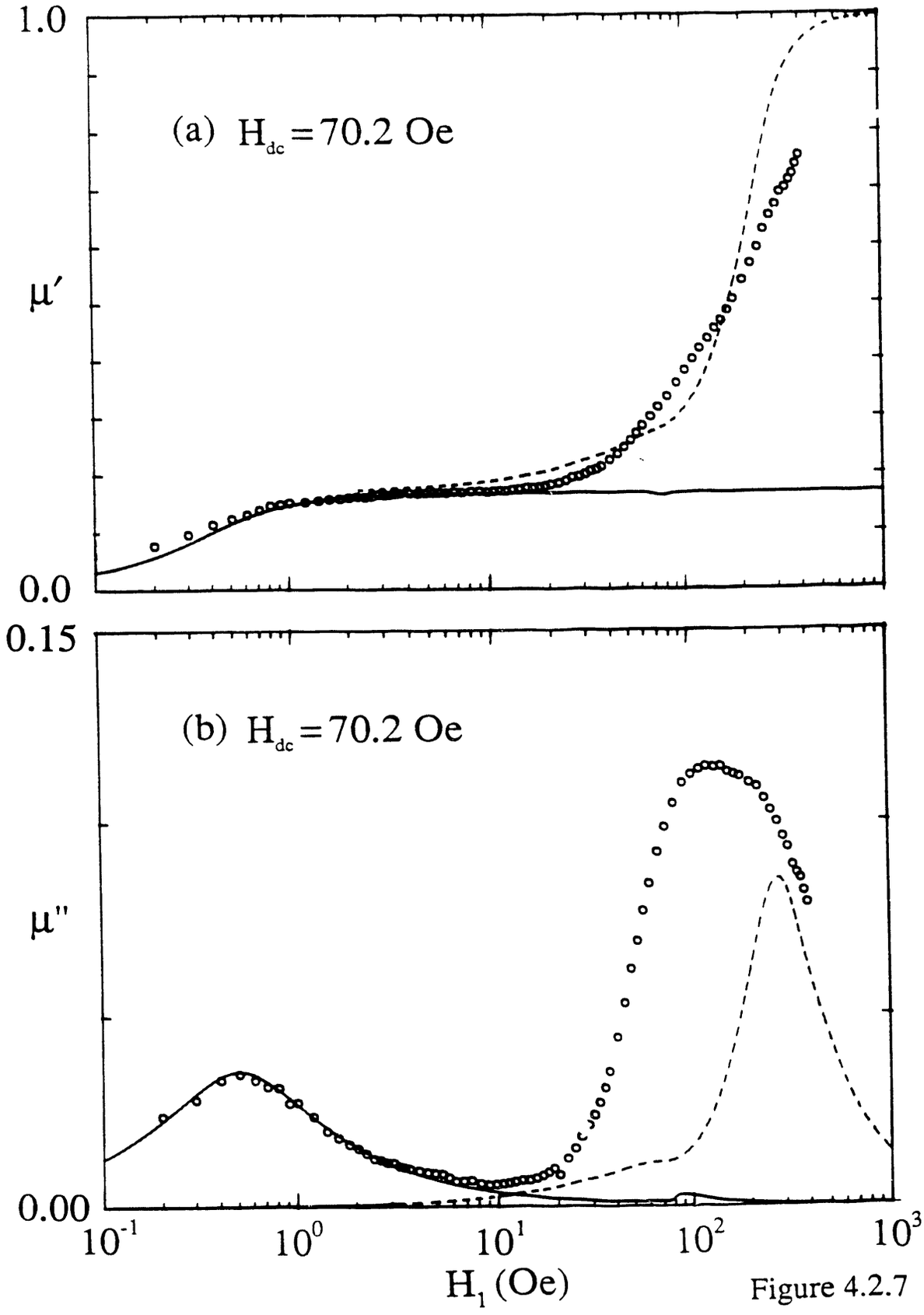


Figure 4.2.7

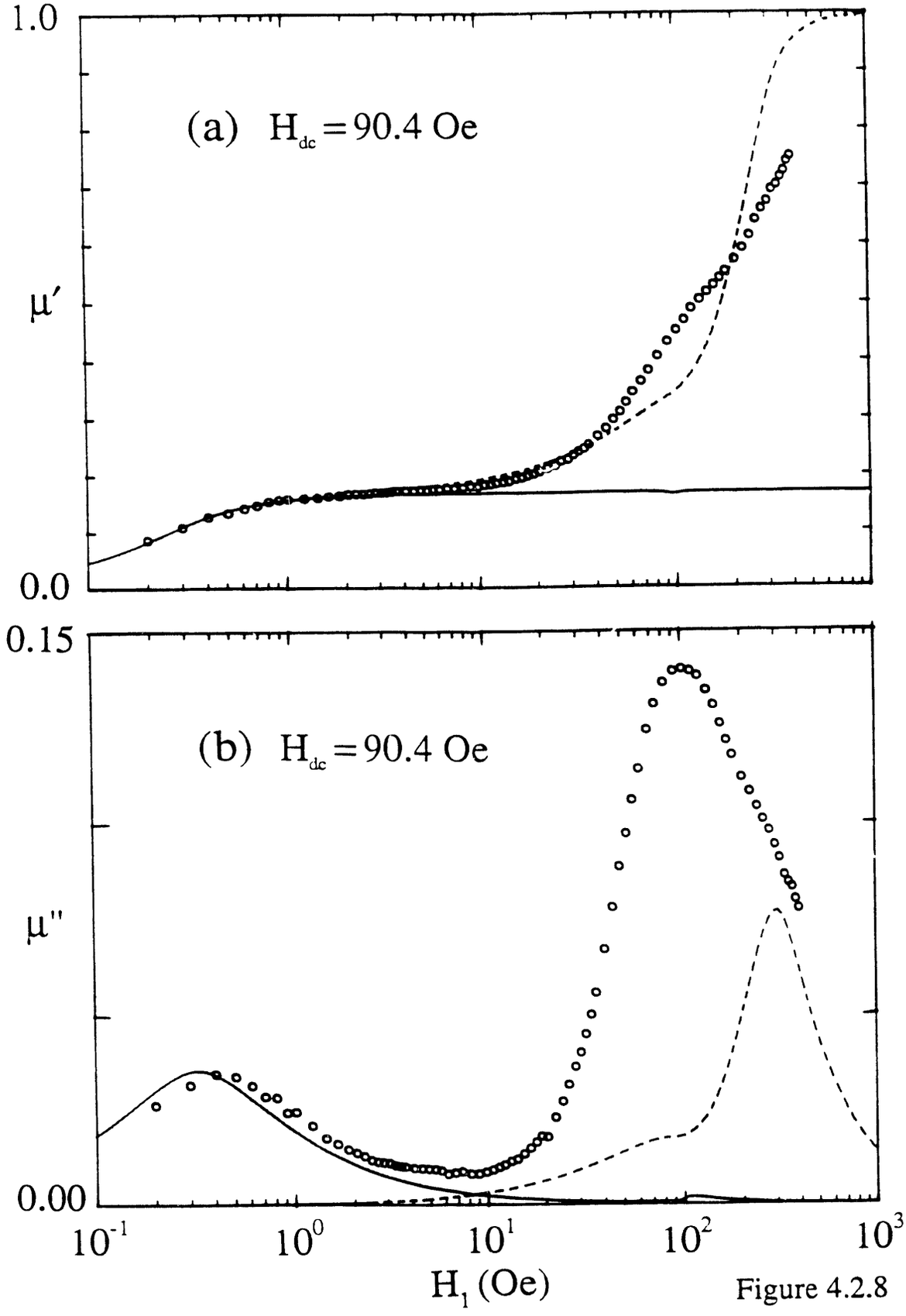


Figure 4.2.8

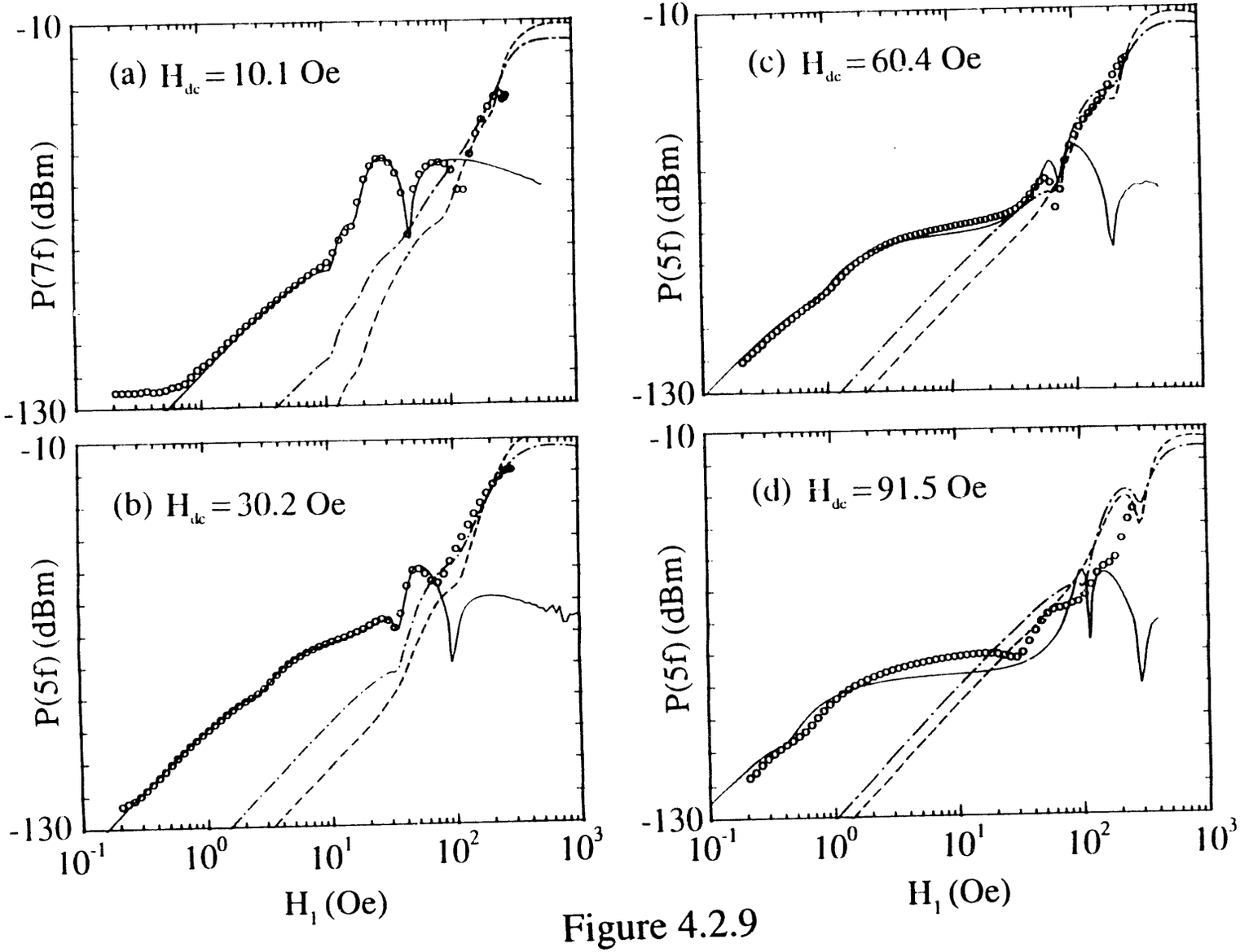

(a) H_{dc} = 10.1 Oe
(b) H_{dc} = 30.2 Oe
(c) H_{dc} = 60.4 Oe
(d) H_{dc} = 91.5 Oe
P(7f) (dBm)
P(5f) (dBm)
-10
-130
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
H_1 (Oe)

Figure 4.2.9

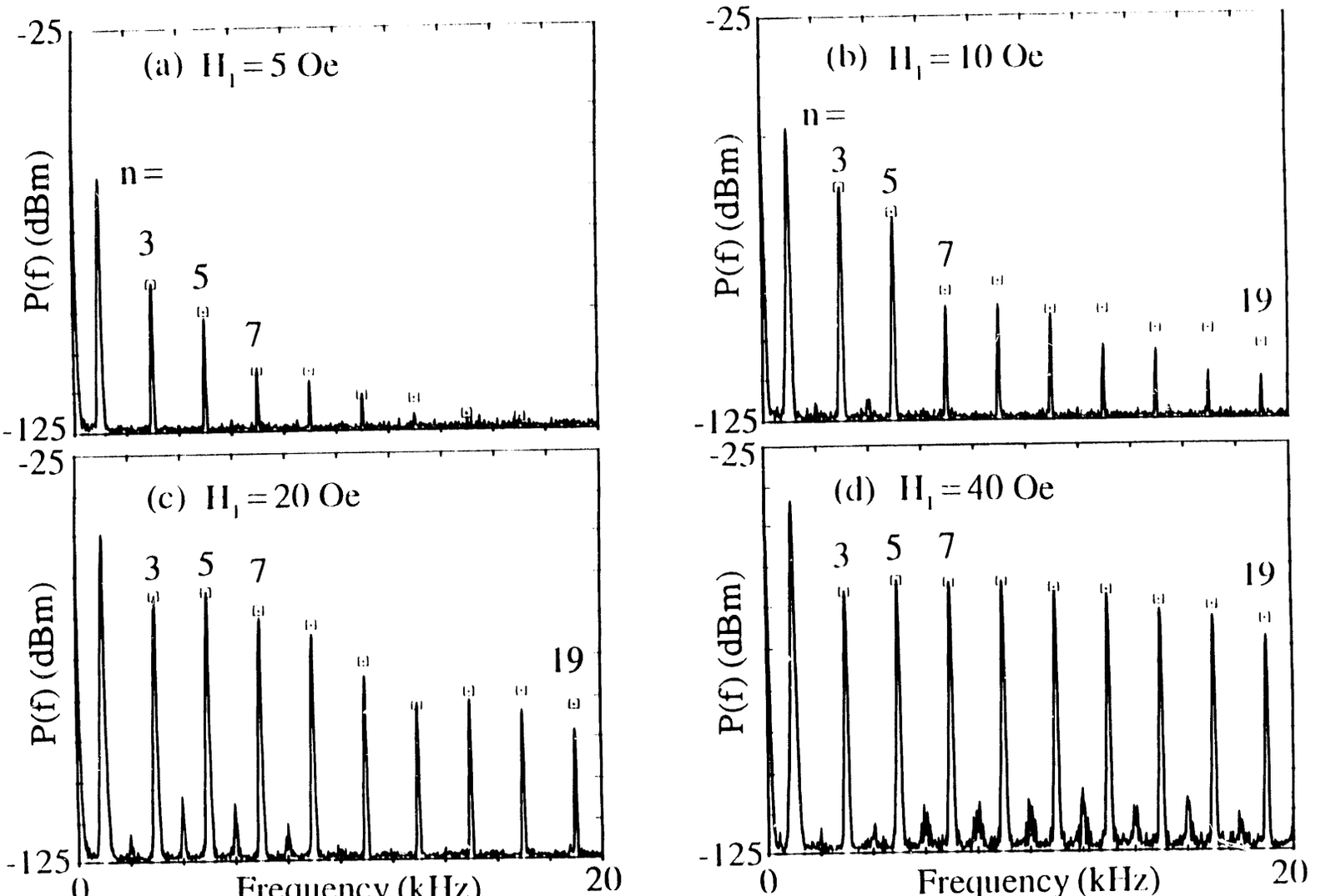

(a) $H_1 = 5$ Oe
(b) $H_1 = 10$ Oe
(c) $H_1 = 20$ Oe
(d) $H_1 = 40$ Oe
n =
3
5
7
19
-25
-125
P(f) (dBm)
0
20
Frequency (kHz)

Figure 4.2.10

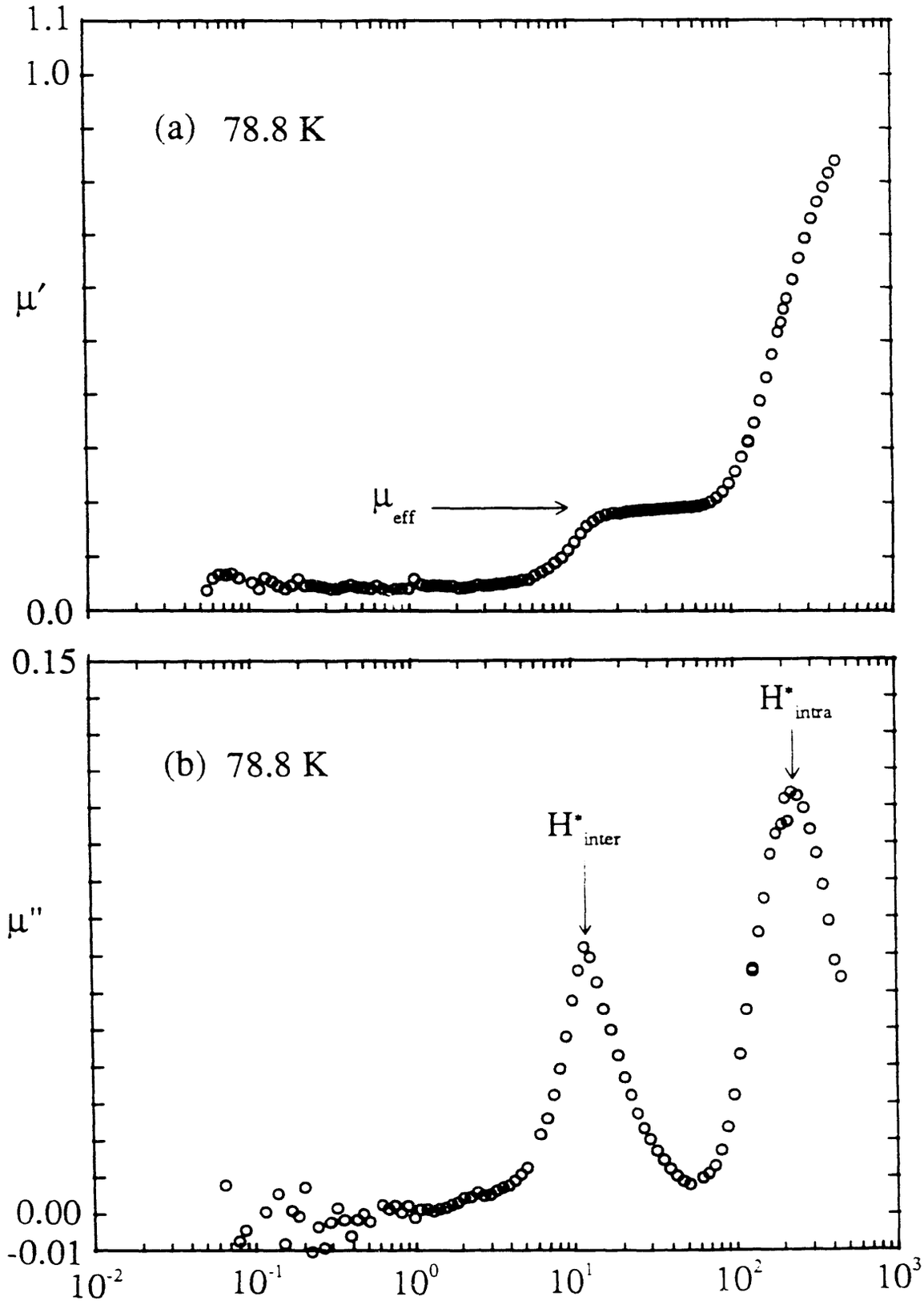

1.1
1.0
(a) 78.8 K
μ′
μ eff
0.0
0.15
H* intra
(b) 78.8 K
H* inter
μ"
0.00
-0.01
10-2
10-1
100
101
102
103
H i (Oe)

Figure 4.2.11

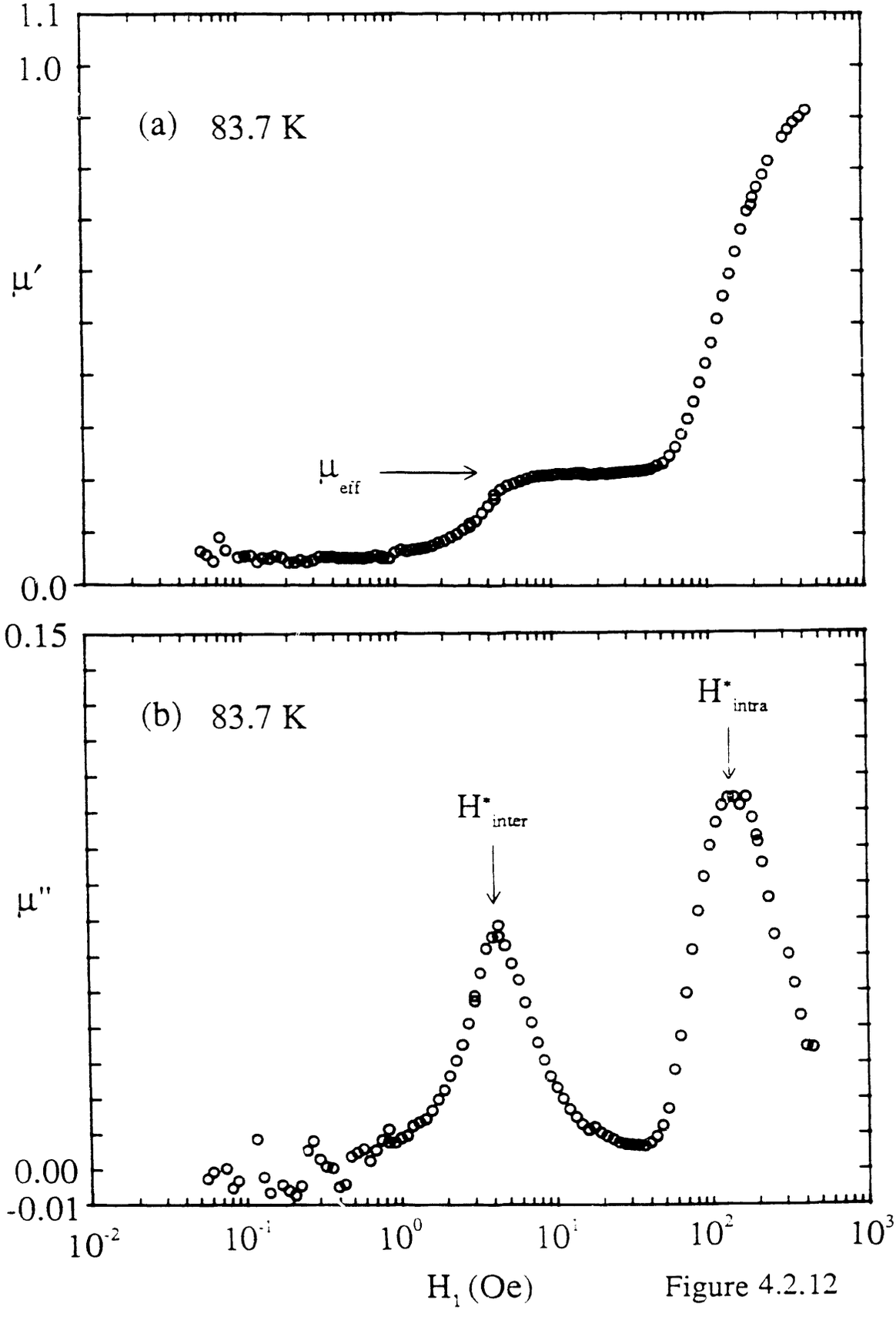

1.1
1.0
(a) 83.7 K
μ′
μ eff
0.0
0.15
(b) 83.7 K
H* intra
H* inter
μ''
0.00
-0.01
10^-2
10^-1
10^0
10^1
10^2
10^3
H1 (Oe)

Figure 4.2.12

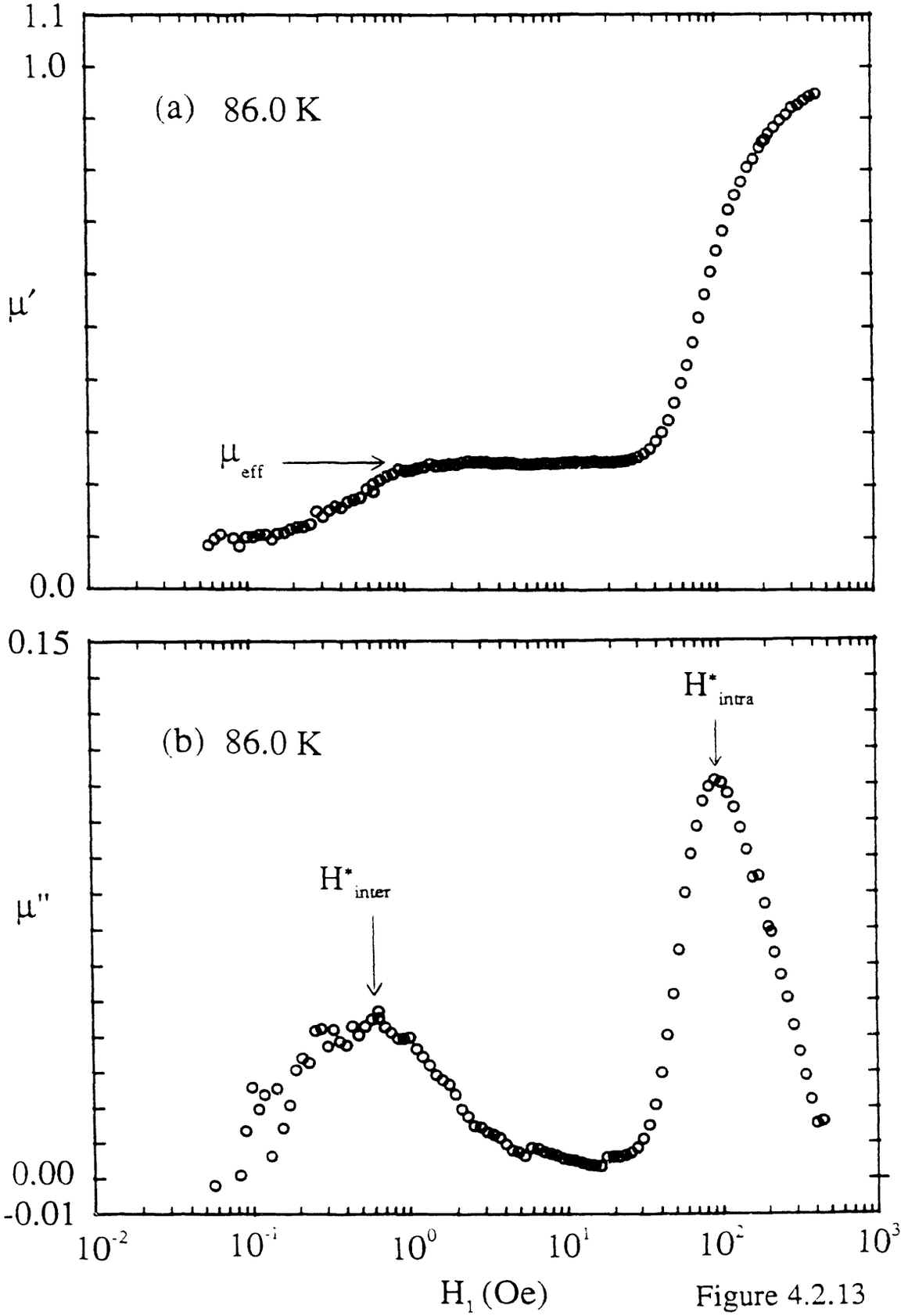

1.1
1.0
(a) 86.0 K
μ′
μ eff
0.0
0.15
(b) 86.0 K
H* intra
H* inter
μ''
0.00
-0.01
10^-2
10^-1
10^0
10^1
10^2
10^3
H1 (Oe)

Figure 4.2.13

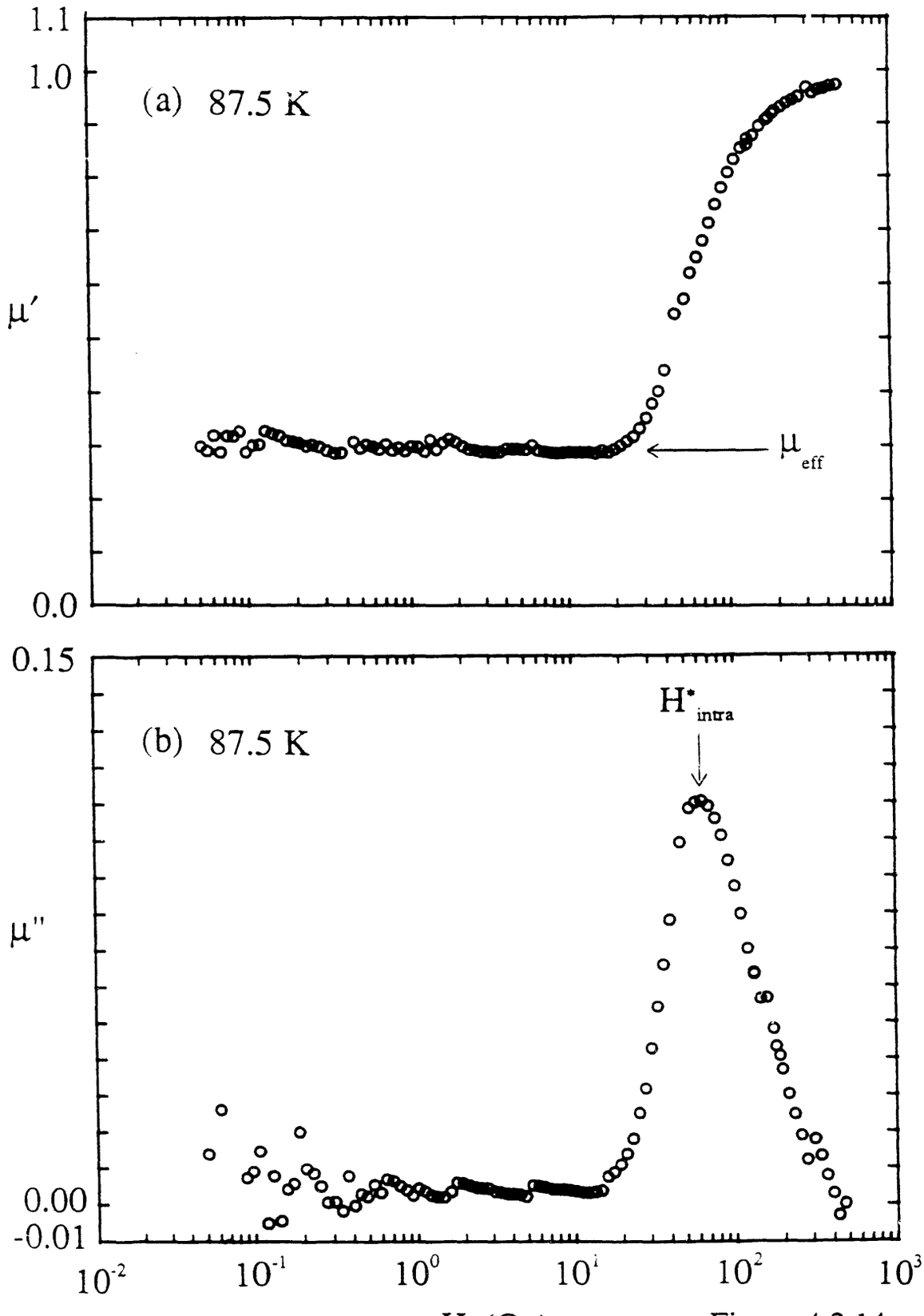

1.1
1.0
(a) 87.5 K
μ′
μ eff
0.0
0.15
H* intra
(b) 87.5 K
μ"
0.00
-0.01
10-2
10-1
100
101
102
103
H1 (Oe)

Figure 4.2.14

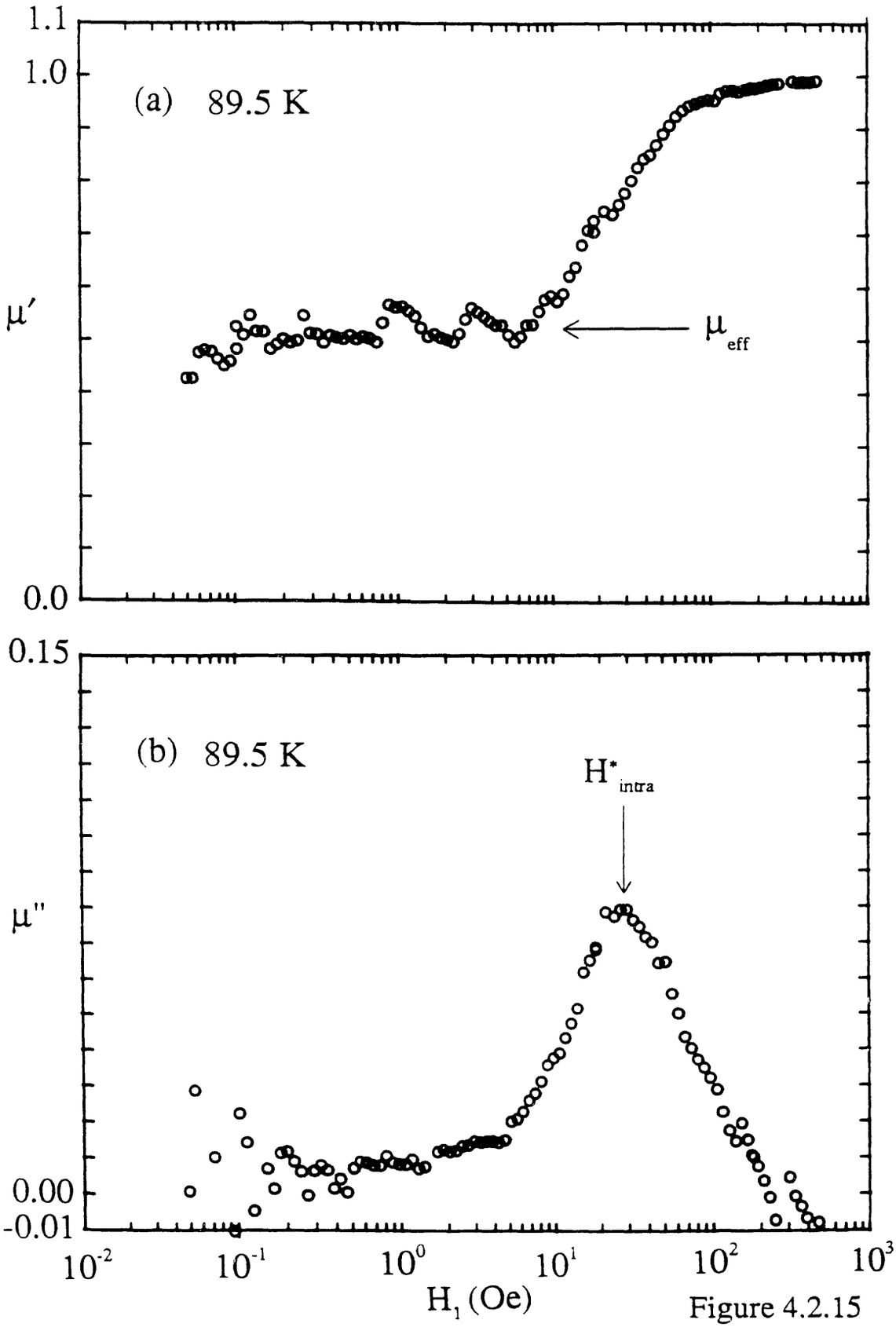

1.1
1.0
(a) 89.5 K
μ′
μ eff
0.0
0.15
(b) 89.5 K
H* intra
μ"
0.00
-0.01
10-2
10-1
100
101
102
103
H1 (Oe)

Figure 4.2.15

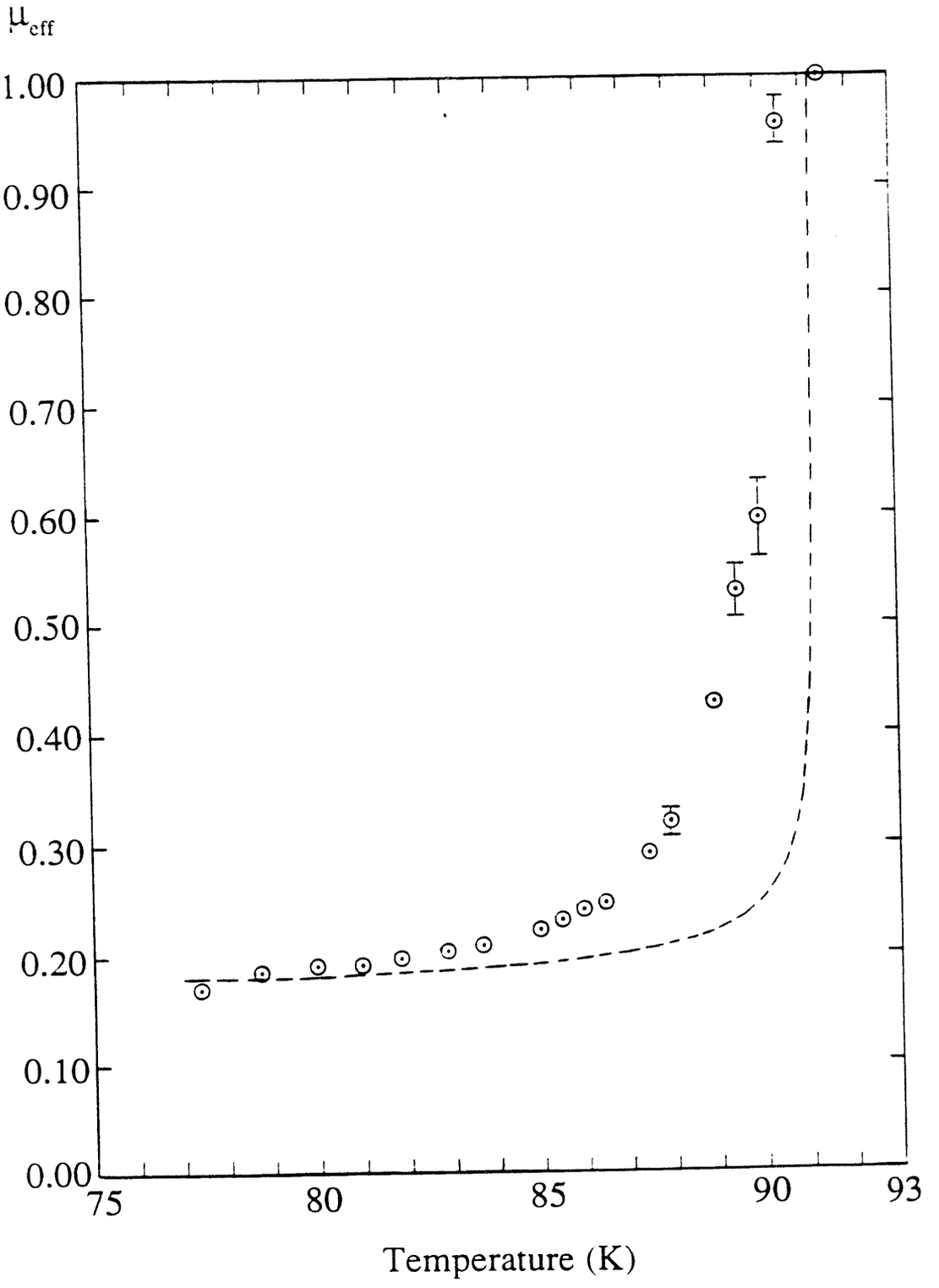

μ_{eff}
1.00
0.90
0.80
0.70
0.60
0.50
0.40
0.30
0.20
0.10
0.00
75
80
85
90
93
Temperature (K)

Figure 4.2.16

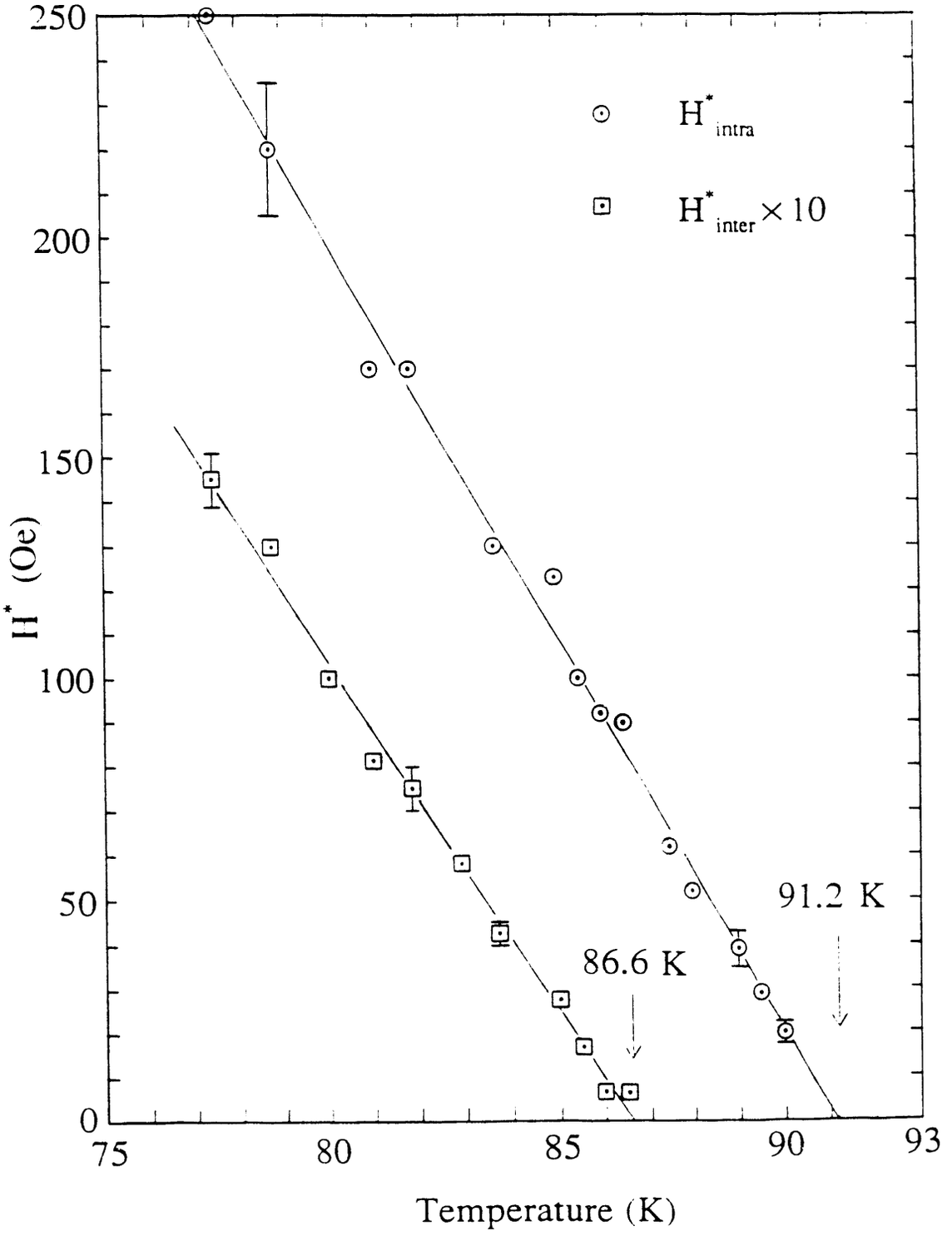

250
200
150
100
50
0
75
80
85
90
93
H* (Oe)
Temperature (K)
H*intra
H*inter×10
91.2 K
86.6 K

Figure 4.2.17

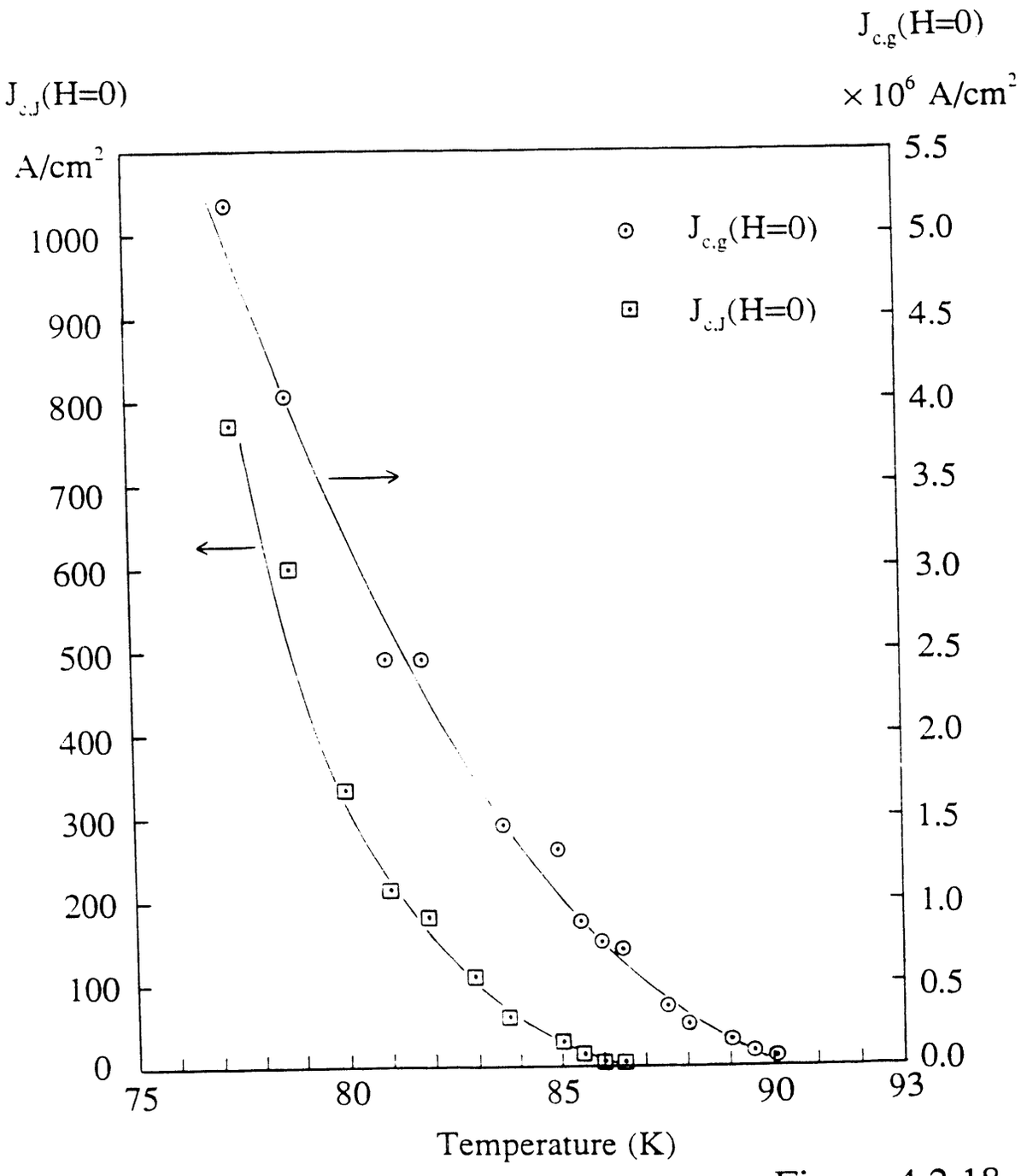

$J_{c,g}$(H=0)
$J_{c,J}$(H=0)
$\times 10^6$ A/cm^2
A/cm^2
1000
900
800
700
600
500
400
300
200
100
0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
$J_{c,g}$(H=0)
$J_{c,J}$(H=0)
75
80
85
90
93
Temperature (K)

Figure 4.2.18

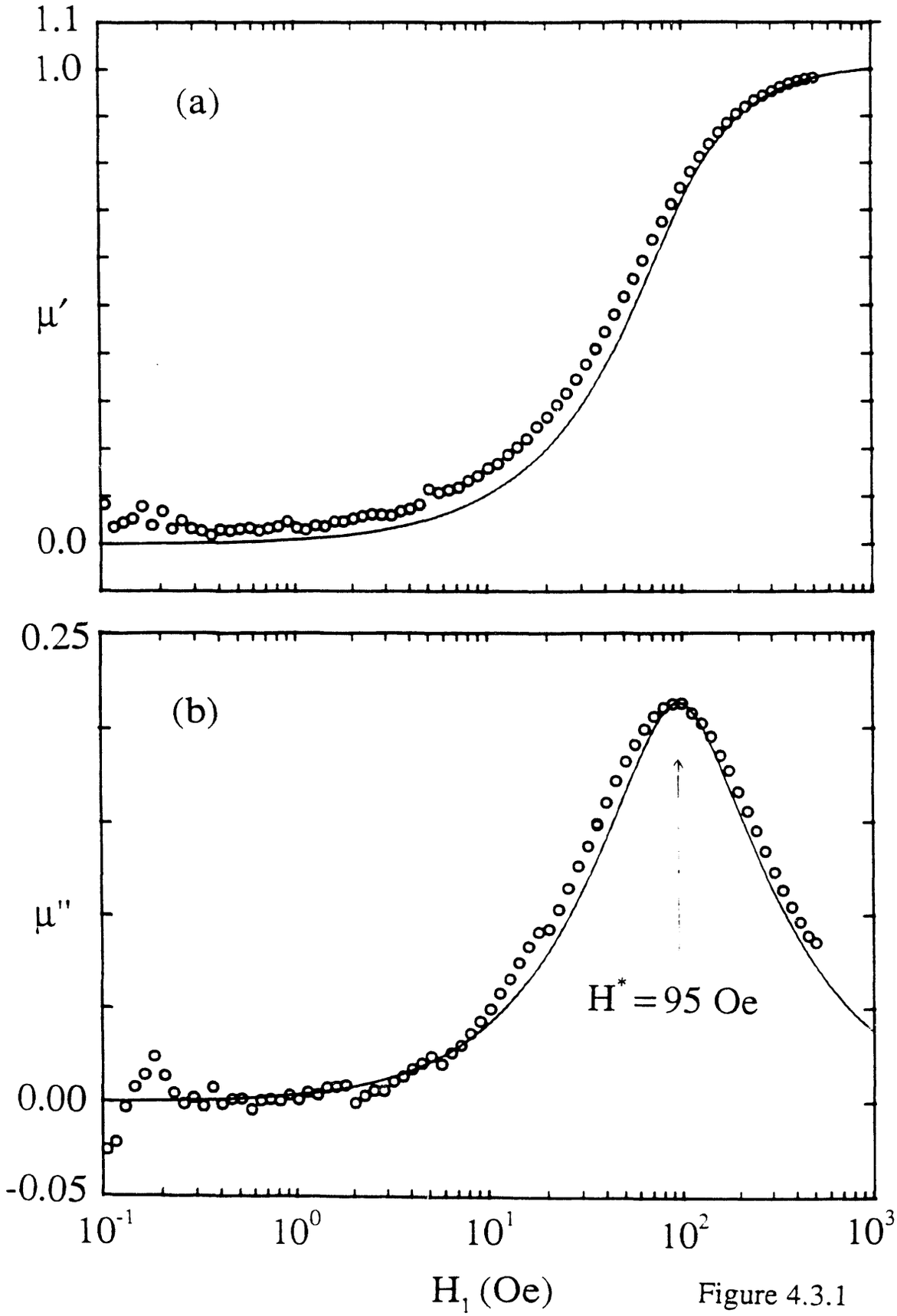


Figure 4.3.1

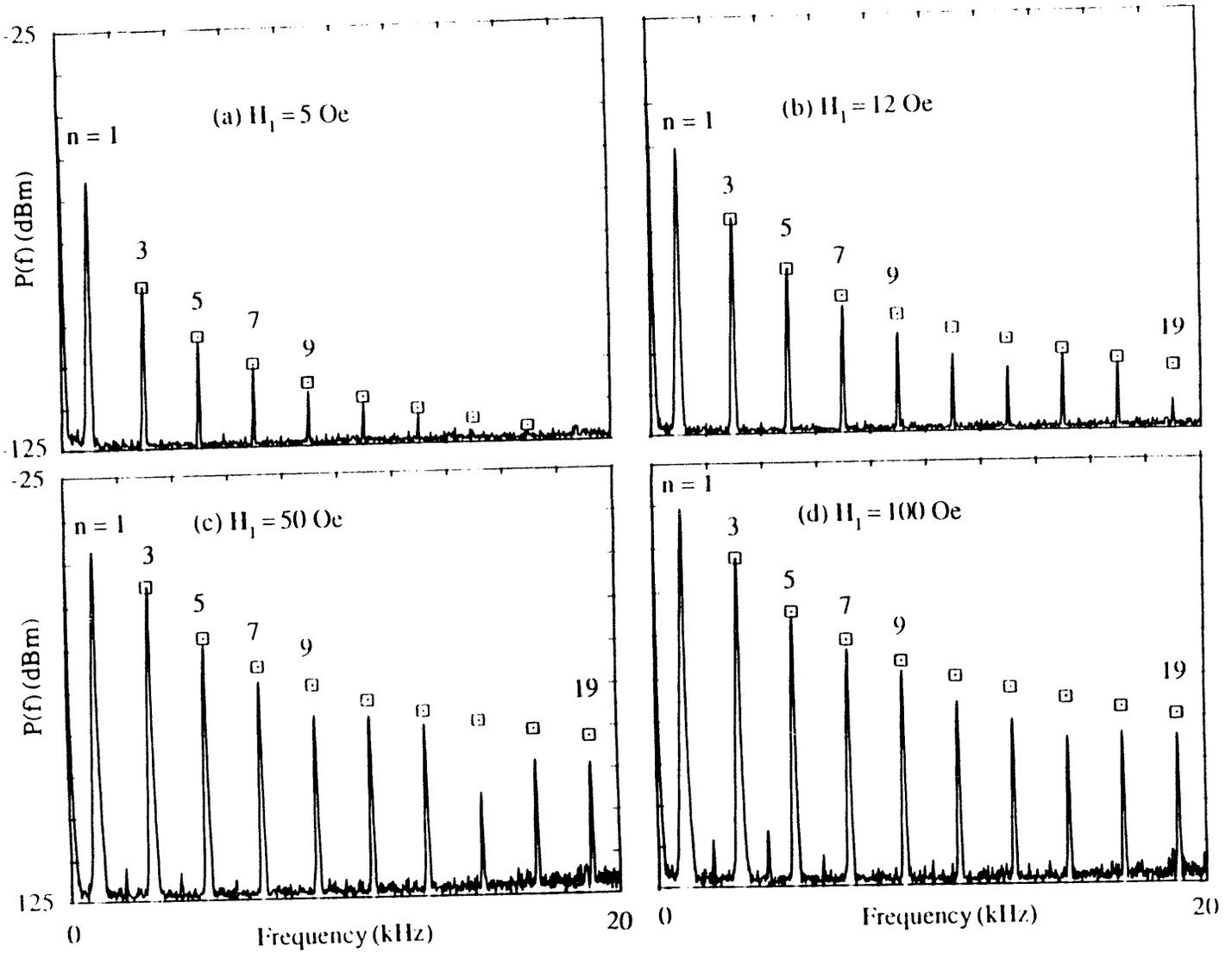

-25
-125
P(f) (dBm)
(a) H_1 = 5 Oe
n = 1
3
5
7
9
(b) H_1 = 12 Oe
n = 1
3
5
7
9
19
-25
125
P(f) (dBm)
n = 1
(c) H_1 = 50 Oe
3
5
7
9
19
n = 1
(d) H_1 = 100 Oe
3
5
7
9
19
0
Frequency (kHz)
20
0
Frequency (kHz)
20

Figure 4.3.2

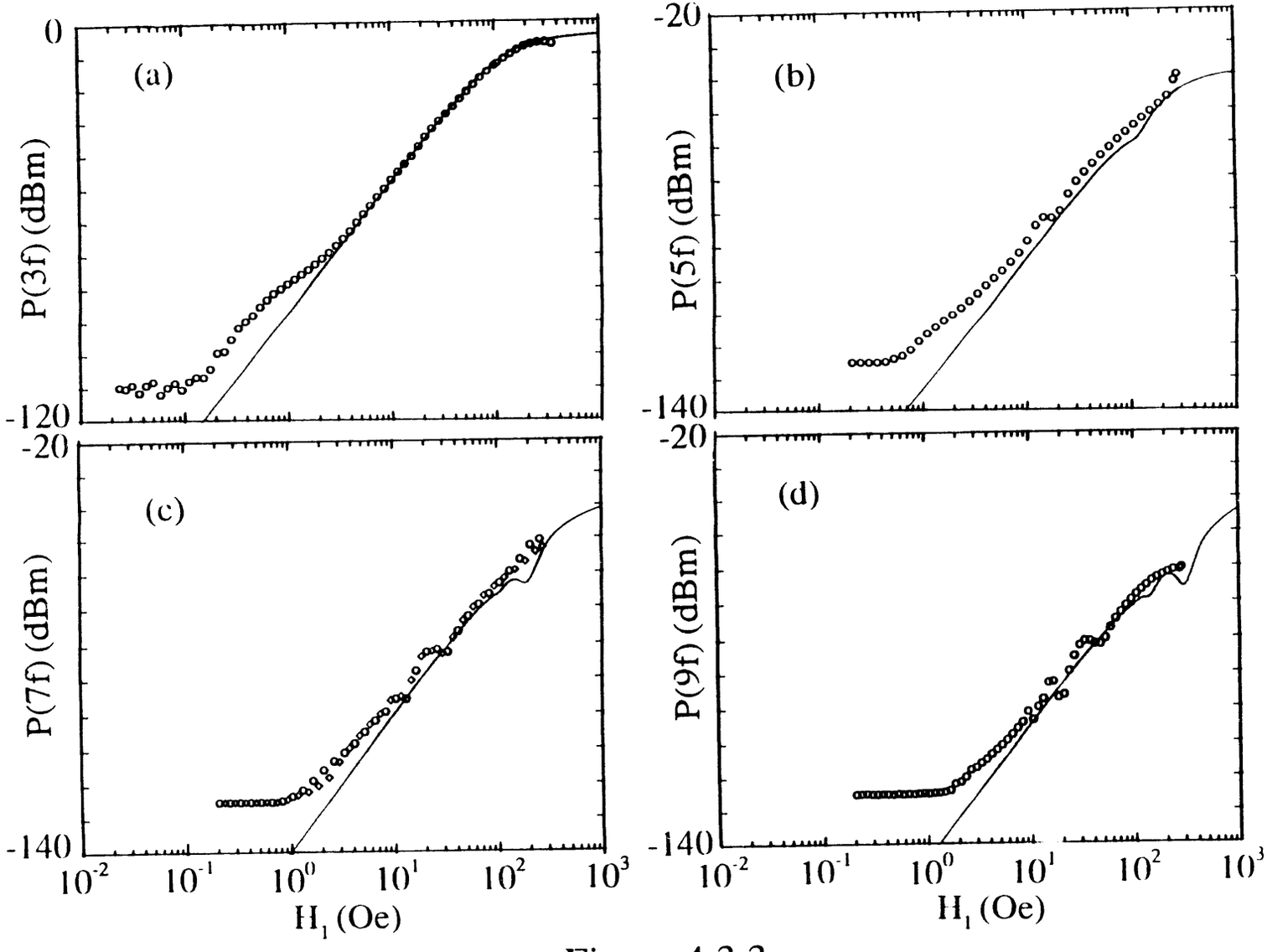

(a)
(b)
(c)
(d)
P(3f) (dBm)
P(5f) (dBm)
P(7f) (dBm)
P(9f) (dBm)
0
-120
-20
-140
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
H_1 (Oe)

Figure 4.3.3

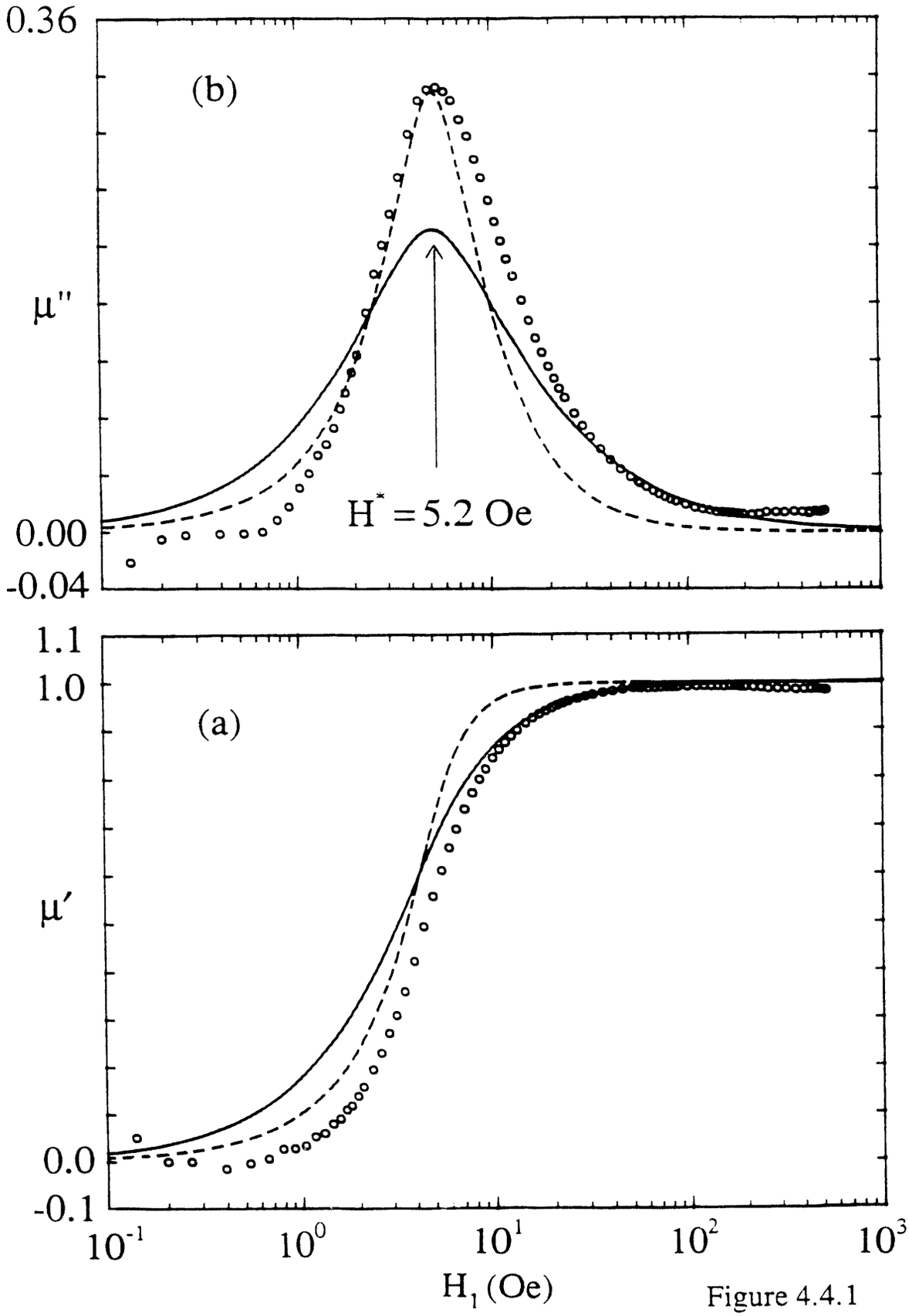


Figure 4.4.1

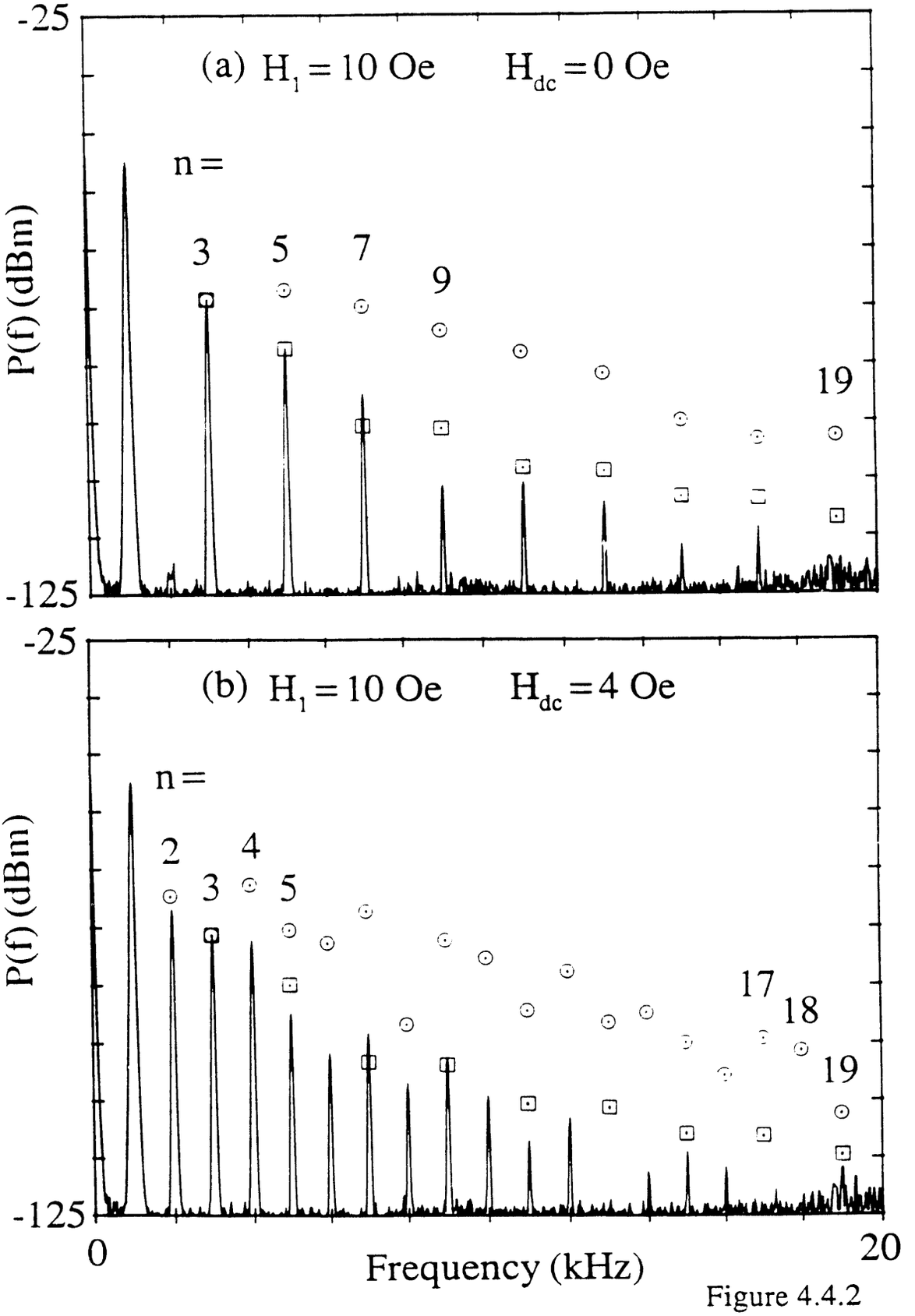

-25
(a) $H_1 = 10$ Oe $H_{dc} = 0$ Oe
n=
3
5
7
9
19
P(f) (dBm)
-125
-25
(b) $H_1 = 10$ Oe $H_{dc} = 4$ Oe
n=
2
3
4
5
17
18
19
P(f) (dBm)
-125
0
Frequency (kHz)
20

Figure 4.4.2

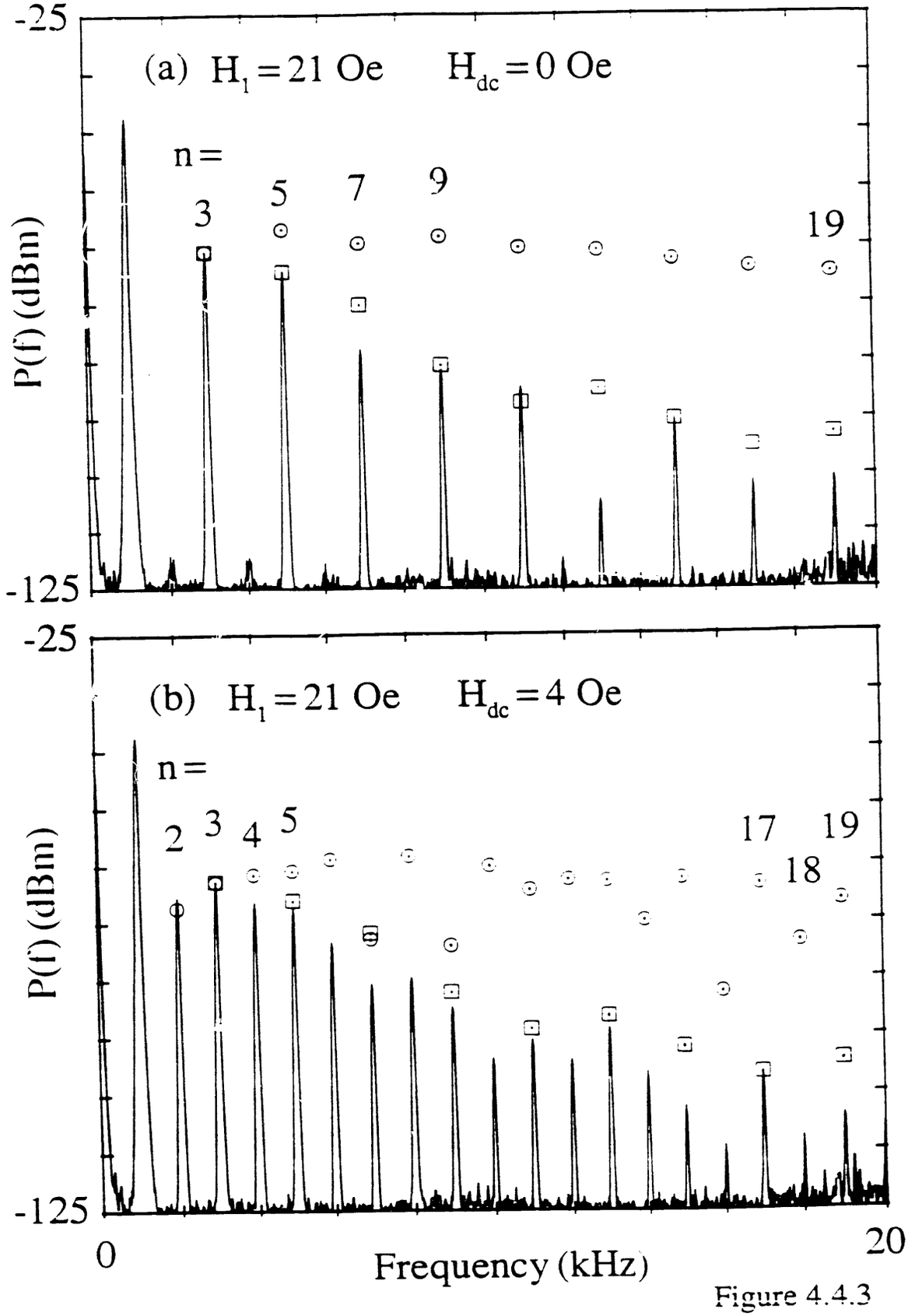

-25
(a) $H_1 = 21$ Oe $H_{dc} = 0$ Oe
n =
3
5
7
9
19
P(f) (dBm)
-125
-25
(b) $H_1 = 21$ Oe $H_{dc} = 4$ Oe
n =
2
3
4
5
17
18
19
P(f) (dBm)
-125
0
Frequency (kHz)
20

Figure 4.4.3

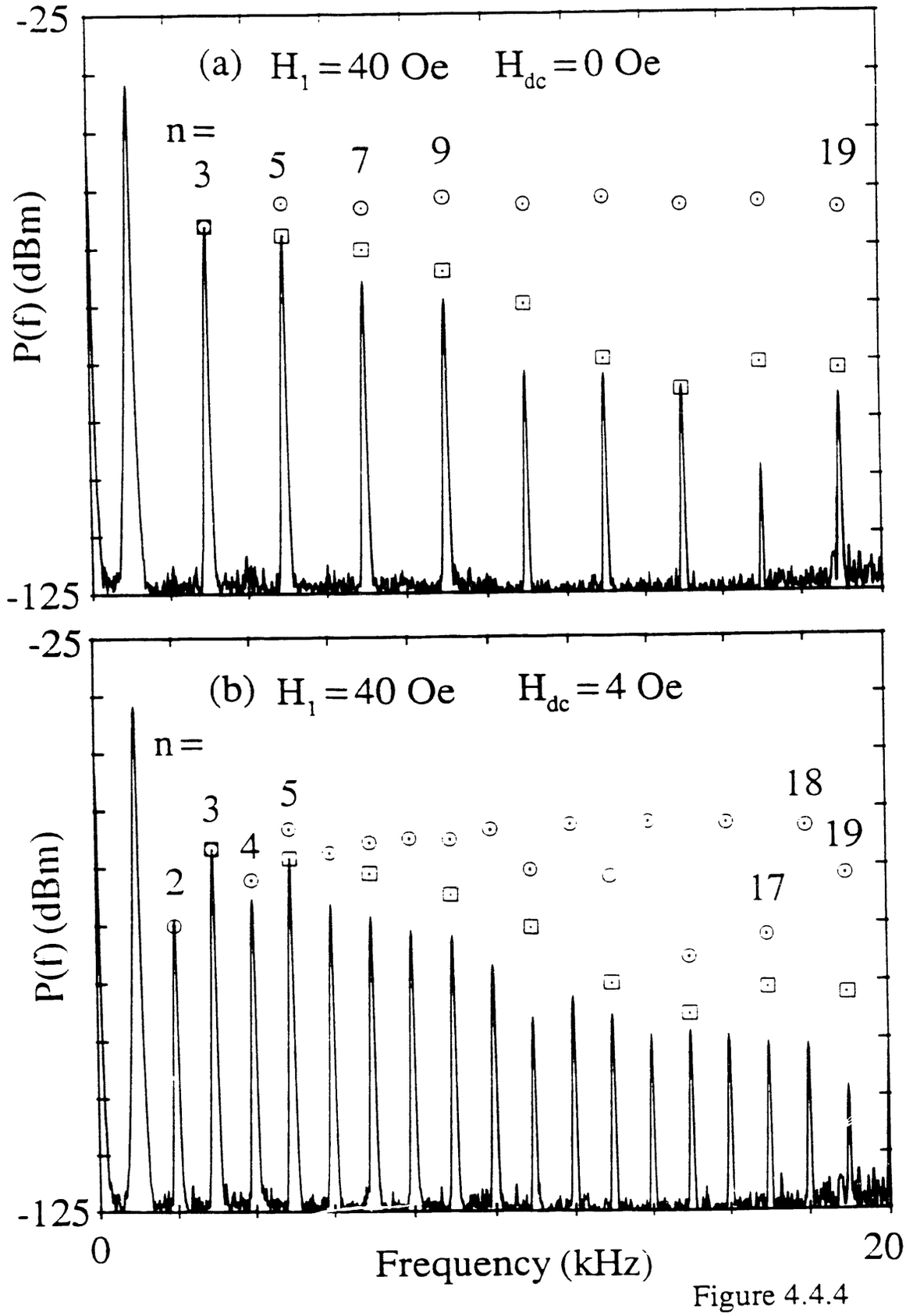

-25
(a) $H_1 = 40$ Oe $H_{dc} = 0$ Oe
n =
3
5
7
9
19
P(f) (dBm)
-125
-25
(b) $H_1 = 40$ Oe $H_{dc} = 4$ Oe
n =
2
3
4
5
17
18
19
P(f) (dBm)
-125
0
Frequency (kHz)
20

Figure 4.4.4

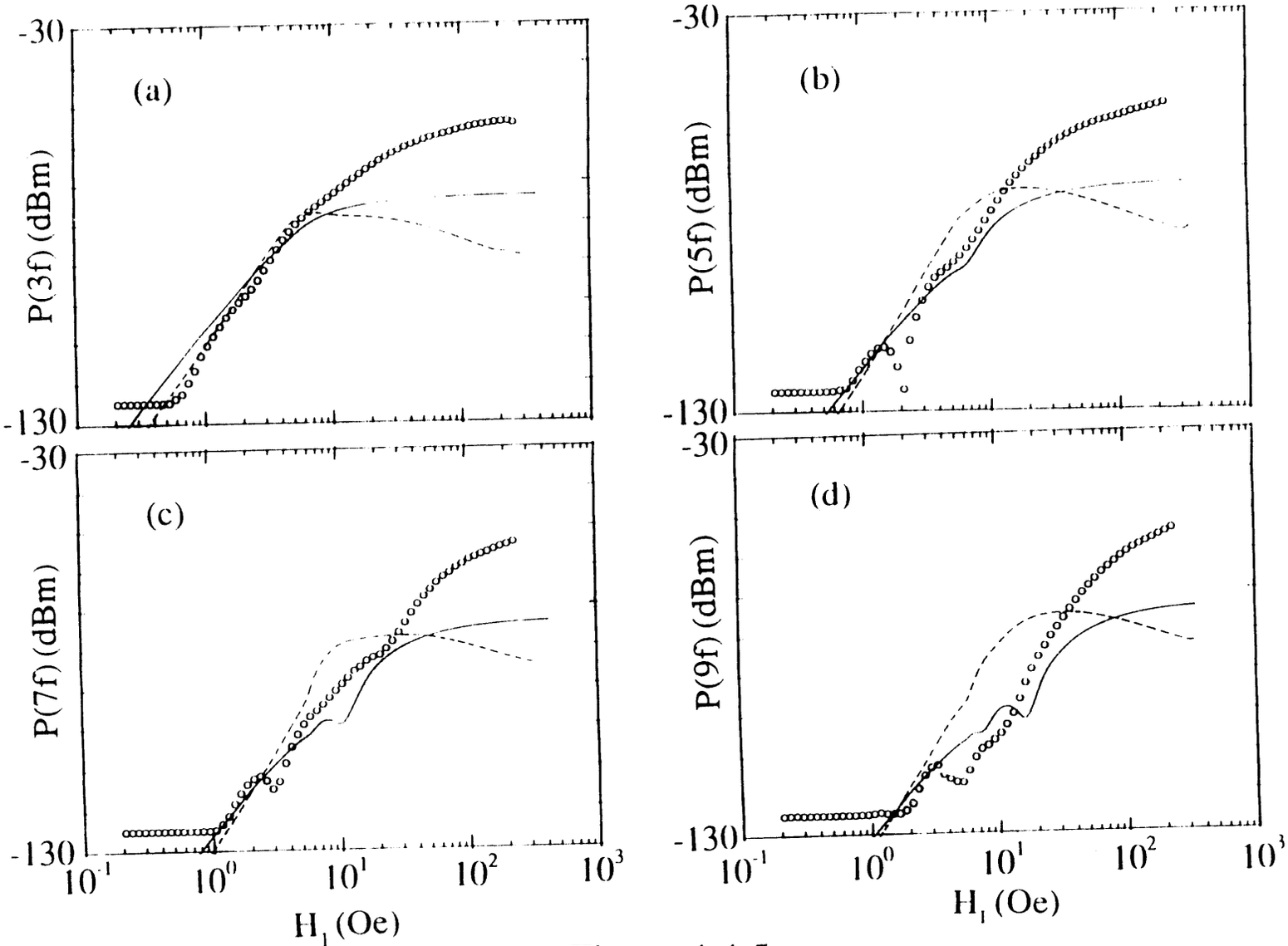

-30
(a)
P(3f) (dBm)
-130
-30
(b)
P(5f) (dBm)
-130
-30
(c)
P(7f) (dBm)
-130
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
H_1 (Oe)
-30
(d)
P(9f) (dBm)
-130
10^{-1}
10^{0}
10^{1}
10^{2}
10^{3}
H_1 (Oe)

Figure 4.4.5

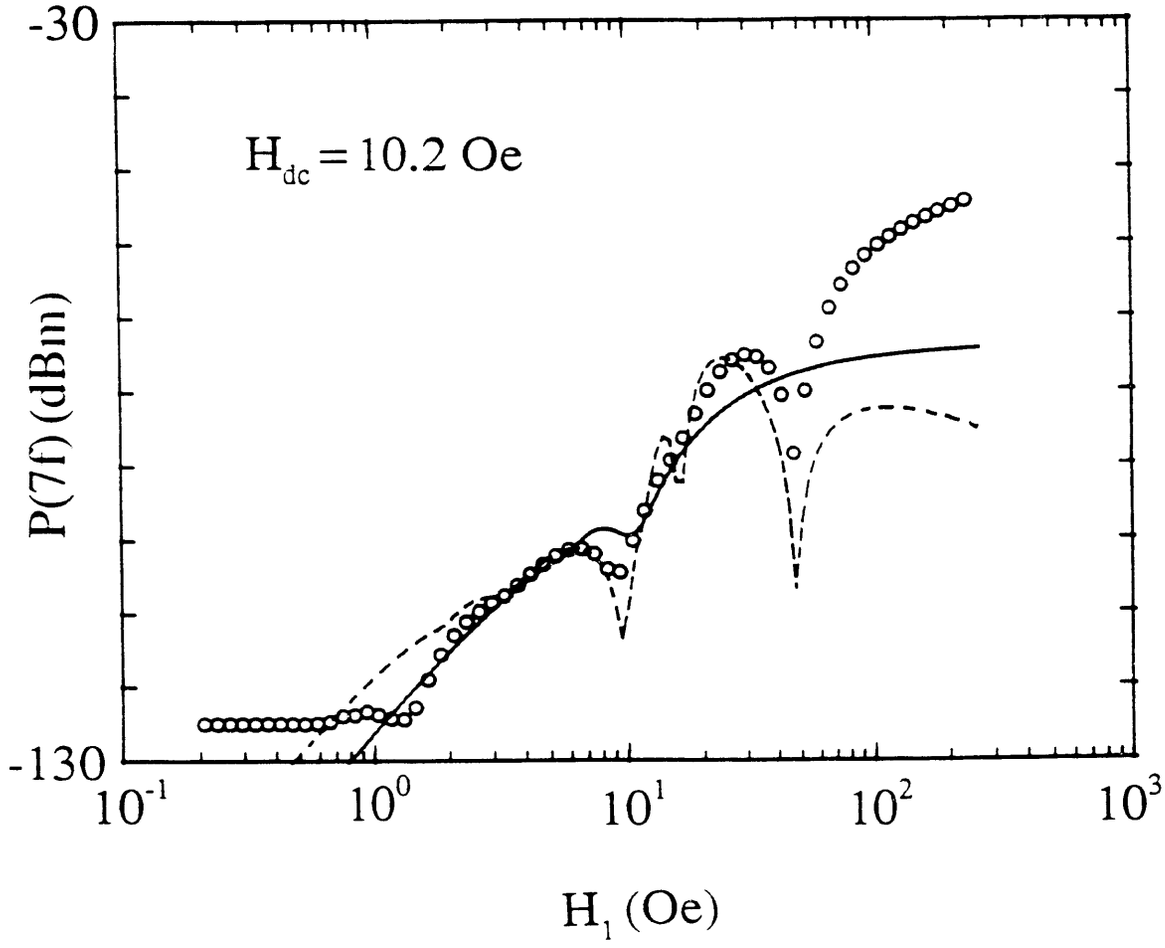


Figure 4.4.6

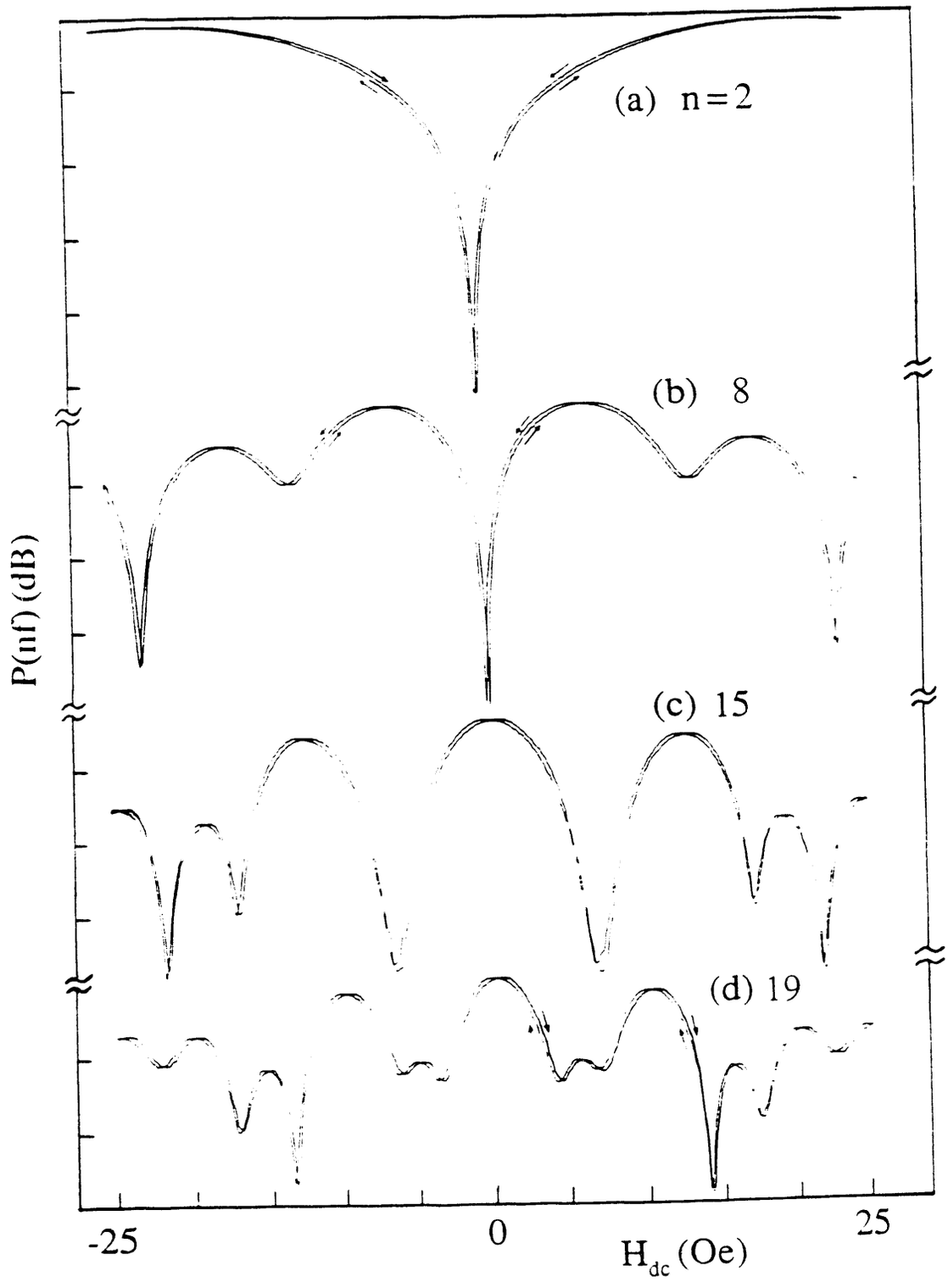


Figure 4.4.7

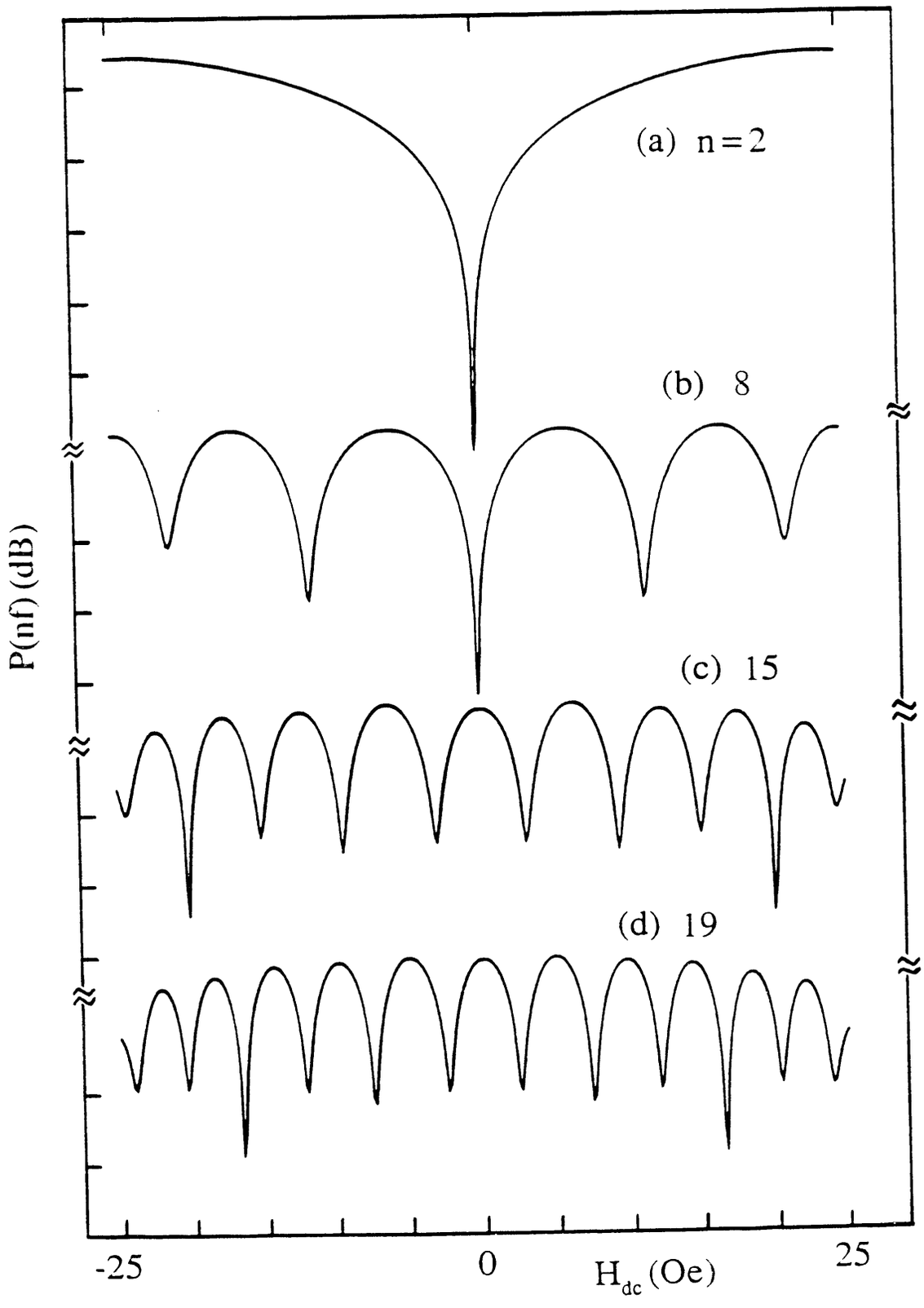


Figure 4.4.8

Chapter 5 Summary and Conclusion

We will briefly summarize in this chapter the most important results of this thesis on nonlinear electrodynamics of high-temperature superconductors.

We presented in Chapter 2 two classes of models of superconducting systems which were found to behave nonlinearly in electrodynamical behavior. The first class of models was suitable for describing superconducting samples in powdered form. In this model, each grain is assumed to behave like a superconducting loop with a Josephson junction, similar to an rf-SQUID. Due to the nonlinear relationship between the loop current and the magnetic flux through the loop, such a prototype superconducting loop was found to generate odd harmonics when driven by an ac magnetic field; if a dc magnetic field is added, even harmonics are also generated. This symmetry of harmonic generation is in agreement with experimental observations. The most surprising result predicted by this model is that, in an ensemble of such prototype superconducting loops, the loop areas o_i which vary broadly, oscillatory dependence on an applied dc magnetic field due to flux quantization in the loops still manifest itself in high enough harmonic response of the system. Naively, one would have expected the wide distribution of loop areas in such an ensemble would smear out the oscillatory behavior of the individual loops; our superconducting loop models predict that the smearing does not occur in the harmonics. Such prediction of pseudo flux quantization behavior is in semi-quantitative agreement with experimental observations.

The second class of models presented in Chapter 2 is based on the Bean-Anderson-Kim critical state model which was originally proposed in the 1960's to explain the dc magnetization hysteresis and flux trapping in conventional low-temperature type-II superconductors. This model, due to its intrinsic hysteretic behavior, is nonlinear in nature and predicts odd harmonic generation when the superconducting sample is driven by an ac magnetic field, with additional even harmonics when a dc field is

added; this is the same as for the first class of models. The original model proposed was generalized here with the addition of one parameter β, essentially amounting to allowing the flux-pinning force density in a superconductor to vary as a power law of magnetic field, $\alpha \sim |H|^{-\beta+1}$. This model was then used to explain experimental results on high-temperature superconducting bulk samples and was found, in the case of ceramic Y-Ba-Cu-O, to fit a variety of data very well quantitatively up to the 10^{th} harmonic, approximately. The essence of the modification is to assume a critical current density of the form $J_c \sim H^{-2}$ for the intergranular medium, which turns out to be close to that measured by transport experiments.

In Chapter 3, details of the experimental apparatus and data acquisition system were given. A table of all the samples involved was also given in this chapter. Relevant software that the author has written to control the experiments was, however, presented in the Appendices.

In Chapter 4, the main results of our experiments on nonlinear electrodynamics and harmonic generation are presented for four different samples. In *powdered Y-Ba-Cu-O*, as predicted by the superconducting-loop models in Chapter 2, approximately periodic oscillations of harmonic power in dc magnetic field were observed. According to the model, the oscillations were due to a pseudo flux quantization of the individual supercurrent loops in the large collection. The dependences of the oscillation "periods" on both the driving ac magnetic field and the harmonic numbers were both found to be as predicted quantitatively by the model. The mathematical and physical reasons that the flux quantization oscillations were not "smeared" out in a collection of loops with a broad area distribution were also presented in this chapter.

Next in Chapter 4 were presented the experimental results on a *ceramic Y-Ba-Cu-O cylinder*. Extensive harmonic generation was also observed in this sample. Generalized critical state model calculations as presented in Chapter 2 were found to fit the harmonic data very well up to the 10^{th} harmonic. Unambiguous experimental

evidence for the *coexistence* of the intergranular and intragranular supercurrents in the same ceramic sample was also presented, together with an estimate of the critical current densities of both supercurrents. The intergranular critical current density was estimated to be about 790 A/cm^2, while the intragranular one was estimated to be about $\sim 10^6$ A/cm^2. When the temperature of the sample was increased while the inter- and intragranular supercurrents were monitored, the intergranular supercurrent was found to go to zero at 86.6 K, while the intragranular one went to zero at 91.2 K. The latter temperature agreed with the critical temperature of the intrinsic superconducting material (Y-Ba-Cu-O), while the former was ascribed to the phase-locking temperature of the 3-dimensional matrix formed by the superconducting grains. Between the phase-locking temperature and the critical-temperature, the individual grains have gone through the superconducting transition, and possess pairing order parameters with non-zero amplitudes. However, due to thermal fluctuations, the phases of their order parameters are randomly oriented and thus incoherent. The result is that even though the grains are individually superconducting, the ceramic as a whole is not; this state is called the paracoherent state. Below the phase-locking temperature, the phases of different grains becomes coherent with one another because now the Josephson coupling energy E_J between the grains is larger than the thermal energy $k_B T$. So, in this coherent state, the whole ceramic becomes superconducting.

Experimental data taken on a pulsed-laser-ablated *Y-Ba-Cu-O thin-film* were presented next in Chapter 4. Only one supercurrent component was observed; no convincing experimental evidence for intergranular supercurrent was found. The data are reasonably fit by a Bean critical state model in which the critical current is not dependent on magnetic fields.

Experimental data on a *Bi-Sr-Ca-Cu-O thin single crystal* were next presented; only one type of supercurrent was found. No model was found that could well explain the harmonic data. Among other possible reasons, this may be due to the geometrical shape

of the sample, giving rise to complicated demagnetization effects.

In conclusion, high-temperature superconductors manifest rich nonlinear electrodynamical behavior. Investigating them through both low-frequency harmonic generation and complex ac permeability provide very severe tests to models and reveals information which would be difficult to acquire with other experimental techniques.

REFERENCES

[1] J. G. Bednorz and K. A. Müller. *Zeitschrift für Physik B*, 64:189, 1986.

[2] H. Küpfer, I. Apfelstedt, W. Schauer, R. Flükiger, R. Meier-Hirmer, and H. Wühl. *Zeitschrift für Physik*, 69:159, 1987.

[3] C. D. Jeffries, Q. H. Lam, Y. Kim, L. C. Bourne, and A. Zettl. *Physical Review B*, 37:9840, 1988.

[4] Q. H. Lam and C. D. Jeffries. *Physical Review B*, 39:4772, 1989.

[5] C. D. Jeffries, Q. H. Lam, Y. Kim, C. M. Kim, A. Zettl, and M. P. Klein. *Physical Review B*, 39:11526, 1989.

[6] T. Xia and D. Stroud. *Physical Review B*, 39:4792, 1989.

[7] K. H. Müller, J. C. Macfarlane, and R. Driver. *Physica C*, 158:366, 1989.

[8] L. Ji, R. H. Sohn, G. C. Spalding, C. J. Lobb, and M. Tinkham. *Physical Review B*, 40:10936, 1989.

[9] Q. H. Lam, Y. Kim, and C. D. Jeffries. *Physical Review B*, 42:4846, 1990.

[10] A. M. Portis. *Electrodynamics of High Temperature Superconductors*. World Scientific, 1991.

[11] J. C. Phillips. *Physics of High-T_c Superconductors*. Academic Press, San Diego, 1989.

[12] C. P. Poole, Jr., T. Datta, and H. A. Farach. *Copper Oxide Superconductors*. Wiley-Interscience, 1988.

[13] M. K. Wu, J. R. Ashburn, C. J. Torng, P. H. Hor, R. L. Meng, L. Gao, Z. J. Huang, Y. Q. Wang, and C. W. Chu. *Physical Review Letters*, 58:908, 1987.

[14] D. E. Farrell, C. M. Williams, S. A. Wolf, N. P. Bansal, and V. G. Kogan. *Physical Review Letters*, 61:2805, 1988.

[15] U. Welp, W. K. Kwok, G. W. Crabtree, K. G. Vandervoort, and J. Z. Liu. *Physical Review Letters*, 62:1908, 1989.

[16] Dong-Ho Wu and S. Sridhar. *Physical Review Letters*, 65:2074, 1990.

[17] T. K. Worthington, W. J. Gallagher, and T. R. Dinger. *Physical Review Letters*, 59:1160, 1987.

[18] S. Sridhar, Dong-Ho Wu, and W. Kennedy. *Physical Review Letters*, 63:1873, 1989.

[19] Y. Tajima, M. Hikita, T. Ishii, H. Fuke, K. Sugiyama, M. Date, A. Yamagishi, A. Katsui, Y. Hidaka, T. Iwata, and S. Tsurumi. *Physical Review B*, 37:7956, 1988.

[20] D. E. Farrell, S. Bonham, J. Foster, Y. C. Chang P. Z. Jiang, K. G. Vandervoort, D. J. Lam, and V. G. Kogan. *Physical Review Letters*, 63:782, 1989.

[21] Y. Hidaka, M. Oda, M. Suzuki, Y. Maeda, Y. Enomoto, and T. Murakami. *Japanese Journal of Applied Physics*, 27:L538, 1988.

[22] R. B. van Dover, L. F. Schneemeyer, E. M. Gyorgy, and J. V. Waszcak. *Physical Review B*, 39:4800, 1989.

[23] J. F. Kwak, E. L. Venturini, D. S. Ginley, and W. Fu. In S. A. Wolf and V. Z. Kresin, editors, *Novel Superconductivity*, page 983, New York, 1987. Plenum Press.

[24] R. L. Peterson and J. W. Ekin. *Physical Review B*, 37:9848, 1988.

[25] D. Esteve, J. M. Martinis, C. Urbina, and M. H. Devoret. *Europhysics Letters*, 3:1237, 1987.

[26] K. A. Müller, M. Takashige, and J. G. Bednorz. *Physical Review Letters*, 58:1143, 1987.

[27] J. R. L. de Almeida and D. J. Thouless. *Journal Physics A*, 11:983, 1978.

[28] C. Ebner and D. Stroud. *Physical Review B*, 31:165, 1985.

[29] J. F. Kwak, E. L. Venturini, P. J. Nigrey, and D. S. Ginley. *Physical Review B*, 37:9749, 1988.

[30] K. W. Blazey, A. M. Portis, K. A. Müller, J. G. Bednorz, and F. H. Holtzberg. *Physica C*, 153-155:56, 1988.

[31] K. W. Blazey, A. M. Portis, K. A. Müller, and F. H. Holtzberg. *Europhysics Letters*, 6:457, 1988.

[32] K. W. Blazey, A. M. Portis, and F. H. Holtzberg. *Physica C*, 157:16, 1989.

[33] Y. Yeshurun and A. P. Malozemoff. *Physical Review Letters*, 60:2202, 1988.

[34] P. W. Anderson. *Physical Review Letters*, 9:309, 1962.

[35] P. W. Anderson and Y. B. Kim. *Review of Modern Physics*, 36:39, 1964.

[36] C. P. Bean. *Review of Modern Physics*, 36:31, 1964.

[37] B. Renker, I. Apfelstedt, H. Küpfer, C. Politis, H. Rietschel, W. Schauer, H. Wühl, U. Gottwick, H. Kneissel, U. Rauchschwalbe, H. Spille, and F. Steglich. *Zeitschrift für Physik B*, 67:1, 1987.

[38] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series, and Products*. Academic Press, Orlando, Florida, 1980.

[39] J. R. Clem. *Physica C*, 153-155:50, 1988.

[40] K. H. Müller and A. J. Pauza. *Physica C*, 161:319, 1989.

[41] S. E. Male, J. Chilton, A. D. Caplin, C. N. Guy, and S. B. Newcomb. *Superconducting Science and Technology*, 2:9, 1989.

[42] Youngtae Kim. *Quasiperiodic Transition to Chaos in Ge; and Magnetic Susceptibility of High-T_c Superconductors*. PhD thesis, University of California at Berkeley, 1990.

[43] Y. Kim, Q. H. Lam, and C. D. Jeffries. *Physical Review B*, 43:11404, 1991.

[44] Note that the ac magnetic field is arbitrarily denoted as $H_1 \sin \omega t$ in Section 2.1, whereas it is $H_1 \cos \omega t$ in Sections 2.2 and 3.2.

[45] Y. B. Kim, C. F. Hempstead, and A. R. Strnad. *Physical Review Letters*, 9:306, 1962.

[46] E. Oran Brigham. *The Fast Fourier Transform*. Prentice-Hall, Englewood Cliffs, New Jersey, 1974.

[47] W. H. Press, B. P. Flannery, and S. A. Teukolsky W. T. Vetterling. *Numerical Recipes*. Cambridge University Press, 1986.

[48] A. M. Campbell. *Journal of Physics C*, 2:1492, 1969.

[49] J. R. Clem. Technical Report IS-M280, Ames National Laboratory, Iowa, 1980.

[50] L. C. Bourne, M. L. Cohen, W. N. Creager, M. F. Crommie, A. M. Stacy, and A. Zettl. *Physics Letters A*, 120:494, 1987.

[51] J.-M. Imer, F. Patthey, B. Dardel, W.-D. Schneider, Y. Baer, Y. Petroff, and A. Zettl. *Physical Review Letters*, 62:336, 1989.

[52] W. Y. Shih, C. Ebner, and D. Stroud. *Physical Review B*, 30:134, 1984.

[53] K. W. Blazey, A. M. Portis, and J. G. Bednorz. *Solid State Communications*, 65:1153, 1988.

[54] M. Tinkham and C. J. Lobb. In H. Ehrenreich and D. Turnbull, editors, *Solid State Physics*, volume 42, page 91. Academic Press, New York, 1989.

[55] I. Morgenstern, K. A. Müller, and J. G. Bednorz. *Zeitschrift für Physik B*, 69:33, 1987.

[56] Y. Yeshurun, I. Felner, and H. Sompolinsky. *Physical Review B*, 36:840, 1987.

[57] A. Raboutou, P. Peyral, J. Rosenblatt, C. Lebeau, O. Pena, A. Perrin, C. Perrin, and M. Sergent. *Europhysics Letters*, 4:1321, 1987.

[58] T. C. Halsey. *Physical Review Letters*, 55:1018, 1985.

[59] A. F. Hebard and A. T. Fiory. *Physical Review Letters*, 44:291, 1980.

[60] M. G. Forrester, Hu Jong Lee, M. Tinkham, and C. J. Lobb. *Physical Review B*, 37:5966, 1988.

[61] Michael Tinkham. *Introduction to Superconductivity*. Krieger, 1985.

[62] P. Pellan, G. Doussalin, H. Cortès, and J. Rosenblatt. *Solid State Communications*, 11:427, 1972.

[63] J. Rosenblatt, H. Cortès, and P. Pellan. *Physics Letters*, 33A:143, 1970.

[64] A. Raboutou, P. Peyral, and J. Rosenblatt. In D. U. Gubser, T. L. Francavilla, J. R. Leibowitz, and S. A. Wolf, editors, *Inhomogeneous Superconductors-1979*, page 272. American Institute of Physics, New York, 1980.

[65] I. S. Jacobs and C. P. Bean. In G. T. Rado and H. Suhl, editors, *Magnetism*, volume III, page 271. Academic Press, New York, 1963.

[66] C. Laurent, D. Mauri, E. Kay, and S. S. P. Parkin. *Journal of Applied Physics*, 65:2017, 1989.

[67] A. M. Campbell and J. E. Evetts. *Advances in Physics*, 21, 1972.

[68] M. P. A. Fisher. *Physical Review Letters*, 62:1415, 1989.

[69] D. S. Fisher, M. P. A. Fisher, and D. A. Huse. *Physical Review B*, 43:130, 1991.

[70] A. P. Malozemoff and M. P. A. Fisher. *Physical Review B*, 42:6784, 1990.

[71] H. Dersch and G. Blatter. *Physical Review B*, 38:11391, 1988.

[72] Y. Yeshurun, M. W. McElfresh, A. P. Malozemoff, J. Hagerhorst-Trewhella, J. Mannhart, F. Holtzberg, and G. V. Chandrashekhar. *Physical Review B*, 42:6322, 1990.

[73] M. Daeumling and D. C. Larbalestier. *Physical Review B*, 40:9350, 1989.

Appendix A Details of Generalized Critical State Model

In this appendix, we describe some of the important points in the analysis of the generalized critical state model described in Chapter 2.

We start from the generalized critical state equation, Eqn. (2.2.10):

$$\frac{dH(r)}{dr} = \pm\frac{4\pi\alpha'}{[\,|H(r)| + H_0\,]^{\beta}} , \tag{A.1}$$

where the geometry of the sample is assumed to be long and cylindrical, with radius $R > 0$. The external applied magnetic field is assumed to be of the form

$$H(R) = H_{dc} + H_1 \cos(\omega t) . \tag{A.2}$$

In the following analysis, we always assume that $H_{dc} \geq 0$; the results for $H_{dc} < 0$ can always be obtained by symmetry arguments.

We will provide the mathematical expressions for both the field profiles inside the sample, $H(r)$, at various instants of the ac magnetic field cycle, and the time derivative of the total magnetic flux, $d\Phi/dt$, in the sample.

First of all, for later notational and computational convenience, we define some variables as follows:

$$A_1 = [H_0 + H(R)]^{\beta+1} + 4\pi\alpha'(\beta+1)R \tag{A.3a}$$

$$A_2 = [H_0 + H_{dc} + H_1]^{\beta+1} - 4\pi\alpha'(\beta+1)R \tag{A.3b}$$

$$A_3 = [H_0 - H(R)]^{\beta+1} - 4\pi\alpha'(\beta+1)R \tag{A.3c}$$

$$A_4 = 2H_0^{\beta+1} - A_3 \tag{A.3d}$$

$$A_5 = [H_0 - (H_{dc} - H_1)]^{\beta+1} - 4\pi\alpha'(\beta+1)R \tag{A.3e}$$

$$A_6 = 2H_0^{\beta+1} - A_5 \tag{A.3f}$$

$$A_7 = [H_0 - H(R)]^{\beta+1} + 4\pi\alpha'(\beta+1)R \tag{A.3g}$$

$$A_8 = [H_0 + H(R)]^{\beta+1} - 4\pi\alpha'(\beta+1)R \tag{A.3h}$$

$$A_9 = 2H_0^{\beta+1} - A_8 \tag{A.3i}$$

$$A_{10} = [H_0 + (H_{dc} - H_1)]^{\beta+1} + 4\pi\alpha'(\beta+1)R \tag{A.3j}$$

$$A_{11} = 2H_0^{\beta+1} - A_2 \,. \tag{A.3k}$$

and

$$B_1 = B_4 = B_6 = B_7 = B_9 = B_{10} = B_{11} = -4\pi\alpha'(\beta+1) \tag{A.4a}$$

$$B_2 = B_3 = B_5 = B_8 = 4\pi\alpha'(\beta+1) \,. \tag{A.4b}$$

From these variables, we further define

$$G_i(r) = A_i + B_i r \,. \tag{A.5}$$

and

$$F_i(r) = \frac{1}{B_i^2}\left[B_i r - (\beta+1)A_i\right]\left[G_i(r)\right]^{\frac{1}{\beta+1}} \,. \tag{A.6}$$

where $i = 1, 2, \ldots, 11$.

Let us also define: (a) r_0 designates the depth of ac field penetration as measured from the axis of the cylindrical sample; (b) r_1 designates the *instantaneous* location of the moving ac flux front; (c) r_2 designates the *instantaneous* location at which the local magnetic field crosses zero and thus changes sign; (d) r_2^0 is the value of r_2 when $H(R) = H_{dc} - H_1$. These variables must of course have values that are greater than or equal to zero. So it is important to note that if the equations provided for them in the following yield negative values for any of these variables, the corresponding variable(s) should be set to zero, ie. the corresponding location concerned have reached the axis of the sample.

A.1 Case I : $H_1 \geq H_{dc} \geq 0$.

For $H_1 \geq H_{dc} \geq 0$:

$$0 \leq r_0 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 - (H_{dc} - H_1)]^{\beta+1} + [H_0 - (H_{dc} - H_1)]^{\beta+1} - 2H_0^{\beta+1} \right\} . \tag{A.1.7}$$

If this expression yields a negative value, then $r_0 \equiv 0$, as mentioned above. Note that in the case of a purely ac applied field, ie. $H_{dc} = 0$, one can derive the penetration field H^* from Eqn. (A.1.7) by setting the right-hand-side expression to zero:

$$0 = r_0 = R - \frac{1}{4\pi\alpha'(\beta+1)} \left\{ [H_0 + H^*]^{\beta+1} - H_0^{\beta+1} \right\} , \qquad \text{(A.1.8)}$$

where the ac field amplitude H_1 has already been replaced in notation by H^*. Eqn. (A.1.8) expresses the fact that when $H_1 = H^*$, the ac flux front reaches the axis of the cylindrical sample. From Eqn. (A.1.8), the expression for H^*, Eqn. (2.2.11), is readily derived.

A.1.1 First half-cycle : $\pi \geq \omega t \geq 0$.

In the first half-cycle of Case I, there are two subcases which have to be considered separately : (i) $H(R) \geq 0$; and (ii) $H(R) \leq 0$.

(i) $H(R) \geq 0$. For $H(R) \geq 0$,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \left\{ [H_0 + (H_{dc} + H_1)]^{\beta+1} - [H_0 + H(R)]^{\beta+1} \right\} . \qquad \text{(A.1.9)}$$

Again, if the above expression yields a negative value, then $r_1 \equiv 0$.

Then for $R \geq r \geq r_1 \geq r_0 \geq 0$:

$$H(r) = -H_0 + [G_1(r)]^{\frac{1}{\beta+1}} . \qquad \text{(A.1.10)}$$

For $r_1 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_2(r)]^{\frac{1}{\beta+1}} . \qquad \text{(A.1.11)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)} [H_0 + H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_1(R) - F_1(r_1) \} . \qquad \text{(A.1.12)}$$

(ii) $H(R) \leq 0$. For $H(R) \leq 0$,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + (H_{dc} + H_1)]^{\beta+1} + [H_0 - H(R)]^{\beta+1} - 2H_0^{\beta+1} \right\}, \quad \text{(A.1.13)}$$

and

$$0 \leq r_2 = R - \frac{1}{4\pi\alpha'(\beta+1)} \left\{ [H_0 - H(R)]^{\beta+1} - H_0^{\beta+1} \right\}. \quad \text{(A.1.14)}$$

If either of these two expressions yields a negative value, the corresponding variable is set to zero.

Then for $R \geq r \geq r_2$:

$$H(r) = H_0 - [G_3(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.15)}$$

For $r_2 \geq r \geq r_1$:

$$H(r) = -H_0 + [G_4(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.16)}$$

For $r_1 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_2(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.17)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)} [H_0 - H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_3(R) - F_3(r_2) + F_4(r_2) - F_4(r_1) \}. \quad \text{(A.1.18)}$$

A.1.2 Second half-cycle : $2\pi \geq \omega t \geq \pi$.

In the second half-cycle, there are three subcases which have to be considered separately: (i) $H(R) \leq 0$; (ii) $0 \leq H(R) \leq H_1 - H_{dc}$; (iii) $0 \leq H_1 - H_{dc} \leq H(R)$.

(i) $H(R) \leq 0$. For $H(R) \leq 0$,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 - (H_{dc} - H_1)]^{\beta+1} - [H_0 - H(R)]^{\beta+1} \right\}, \qquad \text{(A.1.19)}$$

and

$$0 \leq r_2^0 = R - \frac{1}{4\pi\alpha'(\beta+1)} \left\{ [H_0 - (H_{dc} - H_1)]^{\beta+1} - H_0^{\beta+1} \right\}. \qquad \text{(A.1.20)}$$

Then for $R \geq r \geq r_1$:

$$H(r) = H_0 - [G_7(r)]^{\frac{1}{\beta+1}}. \qquad \text{(A.1.21)}$$

For $r_1 \geq r \geq r_2^0$:

$$H(r) = H_0 - [G_5(r)]^{\frac{1}{\beta+1}}. \qquad \text{(A.1.22)}$$

For $r_2^0 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_6(r)]^{\frac{1}{\beta+1}}. \qquad \text{(A.1.23)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)} [H_0 - H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_7(R) - F_7(r_1) \}. \qquad \text{(A.1.24)}$$

(ii) $0 \leq H(R) \leq H_1 - H_{dc}$. For $0 \leq H(R) \leq H_1 - H_{dc}$,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + H(R)]^{\beta+1} + [H_0 - (H_{dc} - H_1)]^{\beta+1} - 2H_0^{\beta+1} \right\}, \quad \text{(A.1.25)}$$

and

$$0 \leq r_2 = R - \frac{1}{4\pi\alpha'(\beta+1)} \left\{ [H_0 + H(R)]^{\beta+1} - H_0^{\beta+1} \right\}. \quad \text{(A.1.26)}$$

Then for $R \geq r \geq r_2$:

$$H(r) = -H_0 + [G_8(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.27)}$$

For $r_2 \geq r \geq r_1$:

$$H(r) = H_0 - [G_9(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.28)}$$

For $r_1 \geq r \geq r_2^0$:

$$H(r) = H_0 - [G_5(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.29)}$$

For $r_2^0 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_6(r)]^{\frac{1}{\beta+1}}. \quad \text{(A.1.30)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)} [H_0 + H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_8(R) - F_8(r_2) + F_9(r_2) - F_9(r_1) \}. \quad \text{(A.1.31)}$$

(iii) $0 \leq H_1 - H_{dc} \leq H(R)$. For $0 \leq H_1 - H_{dc} \leq H(R)$,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + H(R)]^{\beta+1} + [H_0 - (H_{dc} - H_1)]^{\beta+1} - 2H_0^{\beta+1} \right\} . \quad \text{(A.1.32)}$$

Then for $R \geq r \geq r_1$:

$$H(r) = -H_0 + [G_8(r)]^{\frac{1}{\beta+1}} . \quad \text{(A.1.33)}$$

For $r_1 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_6(r)]^{\frac{1}{\beta+1}} . \quad \text{(A.1.34)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)} [H_0 + H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_8(R) - F_8(r_1) \} . \quad \text{(A.1.35)}$$

A.2 Case II : $H_{dc} \geq H_1 \geq 0$.

For Case II :

$$0 \leq r_0 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + (H_{dc} + H_1)]^{\beta+1} - [H_0 + (H_{dc} - H_1)]^{\beta+1} \right\} . \quad \text{(A.2.1)}$$

Note that unlike Eqn. (A.1.7), Eqn. (A.2.1) cannot be used to derive H^* by the very fact of a non-zero H_{dc} in Case II; r_0 denotes the location of the *ac* flux front. However, for a given H_{dc}, eg. $H_{dc} = 10.2$ Oe as in Figure 4.2.2(d), a peak in μ_1'' versus $\log_{10}(H_1)$ will occur approximately when the right-hand-side of Eqn. (A.2.1) goes to zero: $r_0(H_1) \rightarrow 0_+$. This corresponds to the broad peak of μ_1'' at $H_1 = 7.5$ Oe in Figure 4.4.2(d).

A.2.1 First half-cycle : $\pi \geq \omega t \geq 0$.

For Case II, $H(R)$ is always greater than or equal to zero. In the first half-cycle,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + (H_{dc} + H_1)]^{\beta+1} - [H_0 + H(R)]^{\beta+1} \right\} . \quad \text{(A.2.2)}$$

Then for $R \geq r \geq r_1$:

$$H(r) = -H_0 + [G_1(r)]^{\frac{1}{\beta+1}} . \quad \text{(A.2.3)}$$

For $r_1 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_2(r)]^{\frac{1}{\beta+1}} . \quad \text{(A.2.4)}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)}[H_0 + H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_1(R) - F_1(r_1) \} . \quad \text{(A.2.5)}$$

A.2.2 Second half-cycle : $2\pi \geq \omega t \geq \pi$.

In the second half-cycle,

$$0 \leq r_1 = R - \frac{1}{8\pi\alpha'(\beta+1)} \times \left\{ [H_0 + H(R)]^{\beta+1} - [H_0 + (H_{dc} - H_1)]^{\beta+1} \right\} . \quad \text{(A.2.6)}$$

Then for $R \geq r \geq r_1$:

$$H(r) = -H_0 + [G_3(r)]^{\frac{1}{\beta+1}} . \quad \text{(A.2.7)}$$

For $r_1 \geq r \geq r_0$:

$$H(r) = -H_0 + [G_{10}(r)]^{\frac{1}{\beta+1}} . \tag{A.2.8}$$

The time-derivative of the magnetic flux is

$$\frac{1}{2\pi}\frac{d\Phi}{dt} = -\frac{(\beta+1)}{(\beta+2)}[H_0 + H(R)]^{\beta} H_1 \omega \sin(\omega t) \times \{ F_8(R) - F_8(r_1) \} . \tag{A.2.9}$$

Appendix B Computer Programs for Theoretical Calculations

In this appendix, we provide some of the more important computer programs which the author has written and actually used to calculate the various model predictions presented earlier in Chapters 2 and 4.

A list of the main programs used for the "Superconducting Loop Models" presented in Section 2.1 is given in Table B.1. Programs for the generalized critical state model, presented in Section 2.2, are listed in Table B.2. In Table B.3, some general purpose routines that are needed for the some of the programs listed in Tables B.1 and B.2 are also listed. All these programs are given in their complete form later in the appendix.

Table B.1 Programs for "Superconducting Loop Models."

Program name	Model	Purpose(s)
hpjnsincos_avsq_tanh.c	Zero-Order Model	Computes the harmonics generated by an ensemble of superconducting loops as predicted by the "zero-order model" presented in Chapter 2. The results are calculated as a function of the dc magnetic field and plotted immediately by an HP7475A or an HP 7470A plotter.
hp_loop_avsq_tanh.c	Loop Model	The "loop model" version of hpjnsincos_avsq_tanh.c.
RSJ_nr_fftvsh0_avearea.c	First Order Model	Computes the harmonics generated by an ensemble of superconducting loops as predicted by the "first order model" presented in Chapter 2. The results are stored in a data file.

Table B.2 Programs for generalized critical state model.

Program name	Purpose(s)
genKA_spect.c	Computes the harmonic power as predicted by the generalized critical state model as a function of frequency (or harmonic number). The results are given immediately by an HP 7475A or HP 7470A plotter.
genKA_fftvsHa.c	Computes the harmonics generated by a type-II superconductor as predicted by the generalized critical state model presented in Section 2.2. The harmonics are calculated as a function of the ac magnetic field amplitude. The results are stored in a data file.
genKA_fftvsHd.c	Same as genKA_fftvsHa.c, except that the harmonics are calculated as a function of the dc magnetic field.

Table B.3 General purpose routines used for model calculations.

Program name	Purpose(s)
odeint1.c	A Runge-Kutta routine with adaptive stepsize control ("quality-controlled" Runge-Kutta). This routine is needed for the calculations of the loop current's dynamical equation of the "first order model" in program RSJ_nr_fftvsh0_avearea.c.
rkqcrk4.c	A stepper program that takes one "quality-controlled" Runge-Kutta step. This subroutine is needed for odeint1.c.
fftcompon.c	An FFT routine which calculates the harmonic components of the signal voltage as predicted by the "first order model" and, later, the generalized critical state model.

```
/*
**      Program hpjnsincos_avsq_tanh.c
**      This program calculates the harmonics generated by
**      an ensemble of loop/junctions driven by ac magnetic
**      field ("Zero-order model"), and then output the results
**      on an HP plotter.  It multiplies Bessel functions Jn  with
**      sine or cosine, average over areas of the loops, and then
**      square.
**      The averaging is done such that the corresponding
**      dc magnetization has the functional form tanh(x).
**      Ie., the distribution function of the loop areas
**      is
**         sinh[pi.A/(2.sigma)] / {A.[cosh(pi.A/sigma) - 1]}
**
**      This is a preliminary model to see how coupling
**      between Hdc and H1 comes about in the harmonic experiments.
**      The output z-axis is in dB.
**
**      This program was originally written on SUN 3/50 -- UNIX.
**      The graphics routines were originally written by James P. Crutchfield.
*/

#include <math.h>
#define MAX0  501
#define XSPAN1  1.1364   /* size of output graphics */
#define XSPAN2  1.1242   /* size of output graphics */
#define YSPAN  0.94      /* size of output graphics */
#define MAXNUMA 1001
#define PI 3.1415926

main(argc,argv)

int argc;
char *argv[];

{
   double h0[MAX0],h0min,h0max,h0step,h1;              /* h0 is dc field in this program */
   double In,dcterm,acterm[MAXNUMA];
   double ymin,ymax,yshift_step,sh,ch;
   double yshift,xshift,h0span;
   double h1span,Inplot,h0plot,cutoff;
   double posx,posy,xtic,ytic;
   double invsigsq,pAdeltaA[MAXNUMA],norm,Ah0,Ah1;
   double A,probA[MAXNUMA],aveIn,Amin,Amax,Ainterv,sigma;
   int hh1,hh0,N0,order,evenodd,aa,xtraweight,NUMA,plotter;
   int choice,yscale;
   char string[100];

   /* h0min and h0max specify the dc magnetic field range;
     sigma provides a scaling to the characteristic area.
   */
   if (argc != 5)
          {
          printf("Usage\: Command h0min h0max no_of_h0 sigma");
          exit(0);
          }

   sscanf(argv[1],"%lf",&h0min);
   sscanf(argv[2],"%lf",&h0max);
```

```
sscanf(argv[3],"%d",&N0);
if (N0 > MAX0)
        {
        printf("Max. no. of points is %5d.\n",MAX0);
   exit(0);
   }
sscanf(argv[4],"%lf",&sigma);
 printf("Enter Amin, Amax, and no. of A\: ");
 scanf("%lf %lf %d",&Amin,&Amax,&NUMA);
 if (NUMA > MAXNUMA)
         {
         printf("Too many points in A's requested.\n");
         exit(0);
         }
 if (Amin == 0.)
         {
         printf("Amin must be positive, but can be small.\n");
         exit(0);
         }
 Ainterv = (Amax - Amin) / (double)(NUMA - 1);

 printf("Harmonics that you want. \(0,1,2,...\)\n");
 scanf("%d",&order);
 evenodd = order % 2;
printf("Do you want extra weighting of A?");
 printf(" Enter '1' if you do.\n");
 scanf("%d",&xtraweight);

 /* The following prepares a look-up table for averaging
         weighting and normalizing factors. */
 A = Amin;
 norm = 0.;
 for (aa = 0; aa < NUMA; ++aa)
    {
         sh = A * PI * 0.5 / sigma;    sh = sinh(sh);
         ch = A * PI / sigma;                  ch = cosh(ch);

    probA[aa] = sh / (ch - 1.);

    pAdeltaA[aa] = probA[aa] * Ainterv;
    A += Ainterv;
    norm += pAdeltaA[aa];
    }
 norm = 1./norm;  /*normalizing factor, lookup table done*/

 h0step = (h0max - h0min) / (double)(N0-1);

 /* Graphics preparation */
 printf("Which plotter? 1. HP7475A ; 2. HP7470A \: ");
 scanf("%d",&plotter);
 initgraph("/dev/ttya");
 if (plotter == 1)
         window(0.1,0.02,XSPAN1+.1,YSPAN+.02);  /* Set size of hardcopy output */
 else
         window(.08,0.0,XSPAN2+.08,YSPAN);
 color(1);                       /* Choose pen of plotter */

 printf("1. Power in dB  or  2. Amplitude \(linear\) \: ");
 scanf("%d",&yscale);
```

```
if (yscale == 1)
   printf("** If you want vert. scale to be 10 dB/inch, ");

printf("then \(ymax - ymin\) must be 75. **\n");
printf("Enter ymin, ymax, xtic, ytic.\n");
scanf("%lf %lf %lf %lf",&ymin,&ymax,&xtic,&ytic);
cutoff = /* ymin */ -500.;
clear();

h0span = h0max - h0min;
scale(h0min,h0max,ymin,ymax);                    /* Set user's scales w.r.t. plotter's coordinates. */
axes(h0min,ymin,xtic,ytic,1,1);                  /* Draw axes and tick marks. */
axes(h0max,ymax,xtic,ytic,1,1);
border();

posx = h0min + h0span * .10;
posy = ymin + (ymax - ymin) * .95;
sprintf(string,"h0 = \(%6.2lf,%6.2lf\), n = %2d, sig = %4.2lf",
        h0min,h0max,order,sigma);
move(posx,posy);
label(string);     penup();
posy = ymin + (ymax - ymin) * .90;

if (xtraweight == 1)
   sprintf(string,"A = \(%5.3lf, %3.1lf\), wtd, tanh"
                  ,Amin,Amax);
else
   sprintf(string,"A = \(%5.3lf, %3.1lf\), unwtd, tanh"
                  ,Amin,Amax);

move(posx,posy);
label(string);                penup();

for (hh0 = 0; hh0 < N0; ++hh0)
        {
        if (hh0==0)           h0[0] = h0min;
        else                  h0[hh0] = h0[hh0-1] + h0step;
        }

yshift = 0.;
for ( ; )
        {
        if (yscale == 1)
           {
           printf("Enter \(another\) h1 and y-shift\(in dB\), ");
           printf("or ^C if you're done.\n");
           scanf("%lf %lf",&h1,&yshift_step);
           yshift += yshift_step;
           }
        else
           {
           printf("Enter \(another\) h1, and magnification ");
           printf("or ^C if you're done.\n");
           scanf("%lf %lf",&h1,&yshift_step);
           }

        A = Amin;
```

```
        for (aa = 0; aa < NUMA; ++aa)
    {
        Ah1 = A * h1;
        switch(order)
                {
                case 0:
                   acterm[aa] = j0(Ah1);   break;
                case 1:
                   acterm[aa] = j1(Ah1);   break;
          default:
                   acterm[aa] = jn(order,Ah1);  /* Built-in Bessel functions in SUN-Unix */
          }
        A += Ainterv;
        }

  for (hh0 = 0; hh0 < N0; ++hh0)
        {
        A = Amin;
        aveIn = 0.;
        for (aa = 0; aa < NUMA; ++aa)
                {
                Ah0 = A * h0[hh0];
                if (evenodd == 0)
                   dcterm = sin(Ah0); /* even */
                else
                   dcterm = cos(Ah0); /* odd */

           In = acterm[aa] * dcterm;
                In *= pAdeltaA[aa];
                if (xtraweight == 1)
                   In *= A;      /* weighting due to more coupling to coil */

                aveIn += In;
                A += Ainterv;
                }
          aveIn *= norm;

          if (yscale == 1)
                {
                aveIn *= aveIn;
                aveIn = 10. * log10(aveIn);
                Inplot = aveIn + yshift;
                if (Inplot < (yshift + cutoff))
              Inplot = yshift + cutoff;
                }
          else
                {
                aveIn *= yshift_step;
                Inplot = aveIn;
                }

          h0plot = h0[hh0];

          move(h0plot,Inplot);
          if (hh0==0)    pendown();
          }
        penup();
        }
}
```

```
/*
**        Program hp_loop_avsq_tanh.c
**        This program multiplies Bessel functions Jn  with
**        sine or cosine, average, and then square.
**        This is a preliminary model to see how coupling
**        between Hdc and H1 comes about in the harmonic experiments.
**
**        This program is the "Loop Model" version of the program
**        hpjnsincos_avsq_tanh.c; the latter is for the "zero-order
**        model."
**
**    The ensemble averaging is done such that the corresponding
**    dc magnetization has the functional form tanh(x).
**    Ie., the distribution function of the loop areas
**    is
**       sinh[pi.A/(2.sigma)] / {A.[cosh(pi.A/sigma) - 1]}
**
**        The model assumed is a purely superconducting loop without
**        Josephson junction.
*/

#include <math.h>
#define MAX0  501
#define XSPAN1  1.1364    /* size of output graphics */
          #define XSPAN2  1.1242    /* size of output graphics */
#define YSPAN  0.94       /* size of output graphics */
#define MAXNUMA 1001
#define PI 3.1415926
#define MAXEXP  30

main(argc,argv)

int argc;
char *argv[];

{
   /* h0 is the dc field in this program;
     h0min and h0max set the dc field range.
   */
   double expm[MAXEXP],h0[MAX0],h0min,h0max,h0step,h1;
   double In,dcterm,acterm[MAXNUMA];
   double ymin,ymax,yshift_step,sh,ch;
   double yshift,xshift,h0span;
   double h1span,Inplot[MAX0],h0plot,cutoff;
   double posx,posy,xtic,ytic;
             double invsigsq,pAdeltaA[MAXNUMA],norm,Ah0,Ah1;
   double A,probA[MAXNUMA],aveIn,Amin,Amax,Ainterv;
   int hh1,hh0,N0,order,evenodd,aa,xtraweight,NUMA,plotter;
   int choice,yscale,no_expan,nn;
   char string[100];

   if (argc != 4)
          {
          printf("Usage\: Command h0min h0max no_of_h0\n");
          exit(0);
          }

   sscanf(argv[1],"%lf",&h0min);
   sscanf(argv[2],"%lf",&h0max);
```

```
sscanf(argv[3],"%d",&N0);
if (N0 > MAX0)
        {
        printf("Max. no. of points is %5d.\n",MAX0);
  exit(0);
  }
printf("Enter Amin, Amax, and no. of A\: ");
scanf("%lf %lf %d",&Amin,&Amax,&NUMA);
if (NUMA > MAXNUMA)
        {
        printf("Too many points in A's requested.\n");
        exit(0);
        }
if (Amin == 0.)
        {
        printf("Amin must be positive, but can be small.\n");
        exit(0);
        }
Ainterv = (Amax - Amin) / (double)(NUMA - 1);

printf("No. of expansion terms to be included \: ");
scanf("%d",&no_expan);

printf("Harmonics that you want. \(0,1,2,...\)\n");
scanf("%d",&order);
evenodd = order % 2;
printf("Do you want extra weighting of A?");
printf(" Enter '1' if you do.\n");
scanf("%d",&xtraweight);

/* The following prepares a look-up table for averaging
        weighting and normalizing factors. */
norm = 0.;                      A = Amin;
for (aa = 0; aa < NUMA; ++aa)
   {
        sh = A * PI * 0.5;   sh = sinh(sh);
        ch = A * PI;              ch = cosh(ch);

   probA[aa] = sh / (ch - 1.);

   pAdeltaA[aa] = probA[aa] * Ainterv;
   A += Ainterv;
   norm += pAdeltaA[aa];
   }
norm = 1./norm;  /*normalizing factor, lookup table done*/

for (nn = 1; nn <= no_expan; ++nn)
   {
   expm[nn] = -1. * (double)nn;
   expm[nn] += 1.;
   expm[nn] = exp(expm[nn]);
        }

h0step = (h0max - h0min) / (double)(N0-1);

/* Graphics preparation */
printf("Which plotter?  1. HP7475A ;  2. HP7470A \: ");
scanf("%d",&plotter);
```

```
initgraph("/dev/ttya");
if (plotter == 1)
        window(0.1,0.02,XSPAN1+.1,YSPAN+.02);       /* Set size of hardcopy output. */
else
        window(.08,0.0,XSPAN2+.08,YSPAN);
color(1);                       /* Choose pen of plotters. */

printf("1. Power in dB   or   2. Amplitude \(linear\) \: ");
scanf("%d",&yscale);

if (yscale == 1)
        {
  printf("** If you want vert. scale to be 10 dB/inch, ");
        printf("then \(ymax - ymin\) must be 75. **\n");
         }
 printf("Enter ymin, ymax, xtic, ytic.\n");
 scanf("%lf %lf %lf %lf",&ymin,&ymax,&xtic,&ytic);
 cutoff = /* ymin */  -2000.;
 clear();

 h0span = h0max - h0min;
 scale(h0min,h0max,ymin,ymax);   /* Set user's scale w.r.t. plotter's coordinate. */
 axes(h0min,ymin,xtic,ytic,1,1);    /* Draw axes and tick marks. */
 axes(h0max,ymax,xtic,ytic,1,1);
 border();

 posx = h0min + h0span * .10;
 posy = ymin + (ymax - ymin) * .95;
 sprintf(string,"h0 = \(%5.2lf,%5.2lf\), n = %2d, no_expan = %2d",
            h0min,h0max,order,no_expan);
 move(posx,posy);
 /*   label(string); */   penup();
 posy = ymin + (ymax - ymin) * .90;

 if (xtraweight == 1)
    sprintf(string,"A = \(%5.3lf, %3.1lf\), wtd",Amin,Amax);
 else
    sprintf(string,"A = \(%5.3lf, %3.1lf\), unwtd",Amin,Amax);

 move(posx,posy);
 /*   label(string); */                    penup();

 for (hh0 = 0; hh0 < N0; ++hh0)
         {
         if (hh0==0)          h0[0] = h0min;
         else                 h0[hh0] = h0[hh0-1] + h0step;
         }

 yshift = 0.;
 for ( ; ; )
         {
         if (yscale == 1)
            {
            printf("Enter \(another\) h1 and y-shift\(in dB\), ");
            printf("or ^C if you're done.\n");
            scanf("%lf %lf",&h1,&yshift_step);
            yshift += yshift_step;
            }
          else
```

```
    {
    printf("Enter \(another\) h1, and magnification ");
    printf("or ^C if you're done.\n");
    scanf("%lf %lf",&h1,&yshift_step);
    }

  for (hh0 = 0; hh0 < N0; ++hh0)            Inplot[hh0] = 0.;

  for (nn = 1; nn <= no_expan; ++nn)
    {
    A = Amin;
    for (aa = 0; aa < NUMA; ++aa)
          {
          Ah1 = A * h1 * (double)nn;
          switch(order)
            {
            case 0:
                  acterm[aa] = j0(Ah1);   break;
            case 1:
                  acterm[aa] = j1(Ah1);   break;
        default:
                   acterm[aa] = jn(order,Ah1);   /* Built-in Bessel functions in SUN-Unix */
         }
           A += Ainterv;
           }

for (hh0 = 0; hh0 < N0; ++hh0)
           {
           A = Amin;          aveIn = 0.;
           for (aa = 0; aa < NUMA; ++aa)
             {
             Ah0 = A * h0[hh0] * (double)nn;
             if (evenodd == 0)                    dcterm = sin(Ah0); /* even */
             else                                 dcterm = cos(Ah0); /* odd */

        In = acterm[aa] * dcterm * pAdeltaA[aa];
             if (xtraweight == 1)
                  In *= A;/* weighting due to more coupling to coil */

             aveIn += In;      A += Ainterv;
             }
      aveIn *= norm;

           aveIn *= expm[nn];  /* approx. sawtooth expan */

           if ((nn % 2) == 0)  aveIn *= -1.;

           Inplot[hh0] += aveIn;
           }
    }

  for (hh0 = 0; hh0 < N0; ++hh0)
    {
    if (yscale == 1)
          {
          Inplot[hh0] *= Inplot[hh0];
          Inplot[hh0] = 10. * log10(Inplot[hh0]);
          Inplot[hh0] = Inplot[hh0] + yshift;
          if (Inplot[hh0] < (yshift + cutoff))
```

```
                    lnplot[hh0] = yshift + cutoff;
                        }
                    else
                        {
                        lnplot[hh0] *= yshift_step;
                        }

                    h0plot = h0[hh0];

                    move(h0plot,lnplot[hh0]);
                    if (hh0 == 0)    pendown();
                    }
                penup();
                }
        }
```

```
/*
**      Program RSJ_nr_fftvsh0_avearea.c  (First-order model)
**    This is to calculate the AVERAGED (OVER
**      AREAS) time-derivative of the currents in an
**    ensemble of resistance-shunted rf-squids in response to
**    an ac-magnetic field with frequency w=1.
**    Loop areas are distributed as a Gaussian fashion, or as
**      the "sine-transform" of tanh(x):
**
**        F(A) = sinh(A.PI/2) / { A.[ cosh(A.PI) - 1] }
**
**      FFT is done on the time-derivative of the current average,
**    which would be proportional to the 'emf' induced on the
**    pickup coil surrounding the ensemble, and the real and
**      imaginary components are saved.  Finally, these
**      harmonics can be plotted versus H0 (dc magnetic field).
**
**    Additional parameters are the dc-magnetic field,
**    loop inductance and shunt resistance.
**      The model is a single resistance-shunted rf-squid.
**
**        Ac field(dimensionless):   h1 sin(wt)
**        Dc field(     "     ):   h0
**        Loop inductance(  "  ):   L
**        Shunt conductance(  " ):                kap
**        Normal mag. field     :  h = H
**        w (angular frequency): defined to be 1.
**
**        d/dt[Josephson current (normalized by Ic)]
**       = dI/dt
**       = {sin(h0h1sin - LI) - I} / (L.del) + (h1.w/L)cos(wt)
**
**        Has to be linked to odeint1.o, rkqcrk4.o, fftcompon.o.
**
**        Parameters of ode. are given thru external variables.
*/

#include <math.h>
#include <stdio.h>
#include "swgraph.h"
#define  MAX  2048   /*1024 limits N to be <= 10*/
#define  N   11      /*No. of data =  2 to the Nth power*/
#define  JUMP  20    /*N=10 -> JUMP=10; N=9 -> JUMP=5;
                              JUMP = w / winterv= 1 / winterv   */
#define  PI  3.14159265
#define  HPI 1.570796327   /* half Pi */
#define  PI2 6.283185307   /* Pi x 2 */
#define  SAFETY1  0.6
#define  SAFETY2  0.6
#define OMEGA  1.
#define MAXNUMA  501
#define N_HARM 30    /* No. of harmonics to be available */
#define MAX_H0 300     /* Max. no. of h0 within [h0min,h0max]*/

double h0,h1,L,kap,invL,invkap;
int ch;

main(argc,argv)
```

```
int argc;
char *argv[];

{
  /* h0 is the dc magnetic field in this program.
    H0min and H0max set the dc field range.
  */
   extern double h0,h1,L,kap,invkap,invL;
   extern int ch;
   register double *Ij,*Ijdot;
   double t,tmin,tmax,tinterv,t2,snh,cnh;
   double ymin,ymax,ticy,ticx,EPS,dhtry,H0,H1,L0;
   double posx,posy,OMEGAMAX,xmin,xmax;
   double *Avemf,Amin,Amax,Ainterv;
   double *ptr_rl[MAX],*ptr_im[MAX];
   double fft_comp[MAX][2],wmin,wmax,*freq,winterv;
   double *welch,inv_Wss,dummyimag = 0.,norm;
   double sigma,A,probA[MAXNUMA],invsigsq,pAdeltaA[MAXNUMA];
   double H0min,H0max,H0step;
   double comp_rl[N_HARM][MAX_H0],comp_im[N_HARM][MAX_H0];
   double w_har_min[N_HARM],w_har_max[N_HARM],w_harmonic[N_HARM];
   register int ii,aa,ii1;
   int no_data,nodata1,hh1,choice,RSJnrfunc();
   int no_data2,no_data21,AA,no_H0,hh,kk,NUMA,rlorim;
   int wtfunc,xtrawt,AAmin;
   char string[50],filename[20];
   FILE *file_ptr;

   if (argc < 8)
         {
         printf("Usage\: Command H0min H0max no_of_points_in_H0s");
         printf(" h1 L kap sigma \n");
         exit(9);
         }

   sscanf(argv[1],"%lf",&H0min);
   sscanf(argv[2],"%lf",&H0max);
   sscanf(argv[3],"%d",&no_H0);
   sscanf(argv[4],"%lf",&H1);
   sscanf(argv[5],"%lf",&L0);
   sscanf(argv[6],"%lf",&kap);
   sscanf(argv[7],"%lf",&sigma);

   if (no_H0 > MAX_H0)
      {
      printf("Max value of no_H0 = %4d\n",MAX_H0);
      exit(0);
      }

   /* Back-up storage of data */
   printf("Give a name for storage file of results\: ");
   scanf("%s",filename);
   file_ptr = fopen(filename,"w");

   printf("Which weighting function? 1. Gaussian 2. Tanh \: ");
   scanf("%d",&wtfunc);
   printf("Want extra weighting? Enter \'1\' for yes \: ");
   scanf("%d",&xtrawt);
```

```
printf("Enter Amin, Amax, and no. of A\'s\: ");
scanf("%lf %lf %d",&Amin,&Amax,&NUMA);
if (NUMA > MAXNUMA)
   {
   printf("Too many points in A's requested.\n");
   exit(0);
   }
Ainterv = (Amax - Amin) / (double)(NUMA - 1);

if (L0 == 0. || kap == 0.)
   printf("\*\* Have to use small kap version. \*\*\n");
printf("1. Large kap version;  2.  Small kap version\n");
scanf("%d",&ch);
 if (kap != 0.)     invkap = 1. / kap;
 tmin = 0.;
 wmin = 0.;            wmax = 51.2;  /* Nyquist frequency */
 no_data = int_pow(2,N);
 nodata1 = no_data - 1;
 no_data2 = no_data / 2;     no_data21 = no_data2 + 1;
 tinterv = PI / wmax;
 tmax = tinterv * (double)nodata1;
 winterv = 2.*PI / ((double)no_data*tinterv);
 H0step = (H0max - H0min) / ((double)no_H0 - 1.);

 Ij = (double *)calloc(no_data,sizeof(double));
 Ijdot = (double *)calloc(no_data,sizeof(double));
 Avemf = (double *)calloc(no_data,sizeof(double));
 freq = (double *)calloc(no_data21,sizeof(double));

 /*  The following loop prepares for searching the correct peak
 values at various harmonics of the FFT spectrum for each h0.*/
 for (kk = 0; kk < N_HARM; ++kk)
    {
    w_harmonic[kk] = (double)(kk+1);
    /* Window to search for harmonics is set below.*/
    w_har_min[kk] = w_harmonic[kk] - winterv * .2;
    w_har_max[kk] = w_harmonic[kk] + winterv * .2;
    }

 for (ii = 0; ii < no_data; ++ii)
         {
         ptr_rl[ii] = Avemf+ii;     ptr_im[ii] = &dummyimag;
    if (ii <= no_data2)                    *(freq+ii) = winterv * (double)ii;
         }
 /* Welch window, "Numerical Recipes" pp.425. */
 inv_Wss = 0.;
 welch = (double *)calloc(no_data,sizeof(double));
 for (ii = 0; ii < no_data; ++ii)
    {
    *(welch+ii) = (double)ii - .5*((double)no_data-1.);
    *(welch+ii) = *(welch+ii) / (.5*((double)no_data+1.));
    *(welch+ii) = *(welch+ii) * (*(welch+ii))*(-1.);
    *(welch+ii) += 1.;
    inv_Wss += (*(welch+ii) * (*(welch+ii)));
    }
 inv_Wss *= (double)no_data;
 inv_Wss = 1. / inv_Wss;
```

```
* The followings are for making a look-up table of
probability value of A and calculating the normal-
izing factor.  This is done for the averaging. */
if (wtfunc == 1)   invsigsq = 0.5/(sigma*sigma);
norm = 0.;      A = Amin;
for (AA = 0; AA < NUMA; ++AA)
   {
      if (wtfunc == 1)
         {
    probA[AA] = A - 1.;                          /* Gaussian distribution */
    probA[AA] *= probA[AA];
    probA[AA] *= (-1. * invsigsq);
    probA[AA] = exp(probA[AA]); /*unnormalized probability*/
         }
       else
         {
         /* "Sine-transform" of hyperbolic tangent */
         snh = A * PI * 0.5;   snh = sinh(snh);
    cnh = A * PI;        cnh = cosh(cnh);

    probA[AA] = snh / (cnh - 1.);
         }

   pAdeltaA[AA] = probA[AA] * Ainterv;
   norm += pAdeltaA[AA];
   A += Ainterv;
   }
 norm = 1. / norm;   /*normalizing factor*/

 H0 = H0min;
 for (hh = 0; hh < no_H0; ++hh)
   {
       for (ii=0; ii < no_data; ++ii)                *(Avemf+ii) = 0.;

       if (Amin == 0.)
         {
         A = Ainterv;   AAmin = 1;
         }
       else
         {
         A = Amin;   AAmin = 0;
         }

       for (AA = AAmin; AA < NUMA; ++AA)   /* average over areas */
     {
         h0 = A * H0;                  h1 = A * H1;
         L = sqrt(A) * L0;   /*self-induct. assumed proportional to sqrt(A)*/
         if (L != 0.)    invL = 1. / L;

     dhtry = kap * SAFETY1;   /* Time scale of the ode. */
     if (L < 1.)    dhtry *= L;
     if (dhtry > tinterv || dhtry == 0. || ch == 2)      dhtry = tinterv;

     EPS = kap * SAFETY2 / (L + 1.);  /* So that error of Ijdot */
     if (L < 1.)    EPS *= L;       /* is comparable to that of Ij. */
     if (EPS > .01 || EPS==0. || ch==2)
                 EPS = .01;                /* Max. relative error. */
```

```
  t = tmin;     *Ij = 0.;    /*arbit. initial condition*/
        for (ii = 0; ii < nodata1; ++ii)
           {
           ii1 = ii + 1;                        t2 = t + tinterv;
           *(Ij+ii1) = *(Ij+ii);

                /* Adaptive step-size controlled Runge-Kutta step driver. */
           odeint1(Ij+ii1,1,t,t2,EPS,dhtry,RSJnrfunc,Ijdot+ii);

           if (ii1 == nodata1)                          RSJnrfunc(t2,Ij+ii1,Ijdot+ii1);
           t += tinterv;
           }

        for (ii = 0; ii < no_data; ++ii)
           {
                if (xtrawt == 1)  /*due to mutual induct. bet. loop & coil*/
              *(Ijdot+ii) *= A;
           *(Ijdot+ii) *= pAdeltaA[AA];   /*mult. weighting factor*/
           *(Avemf+ii) += *(Ijdot+ii);
           }
         A += Ainterv;
         }

for (ii = 0; ii < no_data; ++ii)     *ptr_rl[ii] *= (*(welch+ii) * norm);

/* fft_comp contains real & imag. components for both
  positive and negative frequencies */
fftcompon(ptr_rl,ptr_im,N,fft_comp);

 /* See Numerical Recipes  Fig.12.2.2 for ranges of ii below.
   See also eqt (12.0.14) of the same book.*/
 for (ii = 0; ii <= no_data2; ii += JUMP)
    {
    for (kk = 0; kk < N_HARM; ++kk)
       {
       /* if within the "window" of the harmonics,...*/
       if (*(freq+ii) >= w_har_min[kk] && *(freq+ii) <= w_har_max[kk])
          {
          comp_rl[kk][hh] = fft_comp[ii][0];
                    comp_im[kk][hh] = fft_comp[ii][1];

          break;    /*The right harmonics is found; no need for
                    further testing of position of freq */
          }
       }
          }
  H0 += H0step;
  }

fprintf(file_ptr,"%6.2lf %4.2lf %4.2lf %4.2lf %5.3lf %3.1lf %4.2lf\n"
                    ,H1,kap,L0,sigma,Amin,Amax,Ainterv);
if (wtfunc == 1)
        {
        if (xtrawt == 1)    fprintf(file_ptr,"Wtd_Gaussian\n");
        else                        fprintf(file_ptr,"unwtd_Gaussian\n");
        }
else
        {
        if (xtrawt == 1)    fprintf(file_ptr,"Wtd, Tanh\n");
```

```
        else                    fprintf(file_ptr,"unwtd, Tanh\n");
        }

fprintf(file_ptr,"%d %d\n",no_H0,N_HARM);

H0 = H0min;
for (hh = 0; hh < no_H0; ++hh)
        {
        fprintf(file_ptr,"%lf  ",H0);
        H0 += H0step;
        }

for (kk = 0; kk < N_HARM; ++kk)
        {
        for (hh=0; hh < no_H0; ++hh)    fprintf(file_ptr,"%lf  ",comp_rl[kk][hh]);
        fprintf(file_ptr,"\n");
        }

for (kk = 0; kk < N_HARM; ++kk)
        {
        for (hh=0; hh < no_H0; ++hh)    fprintf(file_ptr,"%lf  ",comp_im[kk][hh]);
        fprintf(file_ptr,"\n");
        }

fclose(file_ptr);
printf("Data stored.\n");   /* Data stored in file. */

for (;;)
        {
        /* SUN monitor as immediate graphics output */
   initgraph(setdisplaydevice(0,(char **)0));

   printf("Choose the harmonic you want to see\:\n");
   printf("1. Fundamental; 2. 2nd harmonic; ...\n");
   scanf("%d",&choice);
   if (choice > N_HARM)
      {
      printf("Choice not available.\n");
      continue;
      }

        printf("Real or imaginary? \('0' for real\) -- ");
        scanf("%d",&rlorim);

    printf("Input xmin, xmax, ymin, ymax, ticx, ticy resp.\n");
    scanf("%lf %lf %lf %lf %lf %lf",&xmin,&xmax,&ymin,&ymax,
            &ticx,&ticy);
    clear();                    /* Clear the graphics window. */
        window(0.,0.,1.,.8634);                 /* Set size of graphic window. */
    scale(xmin,xmax,ymin,ymax);                 /* Set user's scale w.r.t. window coordinates. */
    border();
    axes(xmin,0.,ticx,ticy,.01,.01);    /* Draw axes and tick marks. */
    axes(xmax,0.,100000.,ticy,.01,.01);
    posx = xmin + (xmax - xmin)*.20;
    posy = ymin + (ymax - ymin)*.96;
    sprintf(string,"n = %2d, h0 = \(%4.1lf,%4.1lf\), h1 = %6.2lf\n",
            choice,xmin,xmax,H1);
    move(posx,posy);        label(string);
    penup();
```

```
    posy = ymin + (ymax - ymin)*.92;
        if (rlorim == 0)
      sprintf(string,"real comp., kap=%4.2lf, L=%4.2lf, sig=%4.2lf"
                  ,kap,L0,sigma);
        else
      sprintf(string,"imag. comp., kap=%4.2lf, L=%4.2lf, sig=%4.2lf"
                  ,kap,L0,sigma);

    move(posx,posy);      label(string);
    penup();
    posy = ymin + (ymax - ymin)*.88;
        if (wtfunc ==1)
           {
           if (xtrawt == 1)
                  {
                  sprintf(string,"A \: %3.1lf  to  %3.1lf, dA = %4.2lf, Wtd Gaussian"
                    ,Amin,Amax,Ainterv);
                  }
           else
                  {
                  sprintf(string,"A \: %3.1lf  to  %3.1lf, dA = %4.2lf, unwtd Gaussian"
                    ,Amin,Amax,Ainterv);
                  }
           }
         else
           {
           if (xtrawt == 1)
                  {
                  sprintf(string,"A \: %5.3lf  to  %3.1lf, dA = %4.2lf, Wtd Tanh"
                    ,Amin,Amax,Ainterv);
                  }
           else
                  {
                  sprintf(string,"A \: %5.3lf  to  %3.1lf, dA = %4.2lf, unwtd Tanh"
                    ,Amin,Amax,Ainterv);
                  }
           }

    move(posx,posy);      label(string);                        penup();

    H0 = H0min;
    for (hh=0; hh < no_H0; ++hh)
       {

           if (rlorim == 0)      move(H0,comp_rl[choice-1][hh]);
           else                        move(H0,comp_im[choice-1][hh]);

       if (hh==0)   pendown();
       H0 += H0step;
       }
     penup();

          printf("Hit <return> to complete this output. ");
          getchar();          getchar();
          exitgraph();
     }
}
```

```
/*
**    Function RSJnrfunc().              ("First-order model")
**    This is the differential equation derived for the
**    time-derivative of the Josephson junction current
**    in a resistance-shunted rf-squid with self-inductance.
**    No approximation has been made for the magnitude
**    of the (dimensionless) inductance.
**
**    dI/dt = {sin(h0h1sin - LI) - I}/(L.del) + (h1.w/L)cos(wt)
**
**    where L is the dimensionless inductance
**       kap is the dimensionless shunt-conductance.
**       h0h1sin = h0 + h1.sin(wt)
**            w (angular frequency) is defined to be 1.
*/

RSJnrfunc(t,I,Idot)

double t,*I,*Idot;

{
  extern double h0,h1,L,kap,invkap,invL;
  extern int ch;                    /*Externs' values won't change*/
  double LI,cost,sint,dummy1,S,C,LC1,h0h1sinLI;

  cost = cos(t);                    sint = sin(t);
  LI = L * (*I);
  h0h1sinLI = h0 + h1*sint - LI;
  S = sin(h0h1sinLI);

  if (ch == 1)
    {
    *Idot = (S - *I) * invkap;
    *Idot += (h1 * cost);
    *Idot *= invL;
    }
  else
    {
        C = cos(h0h1sinLI);
        LC1 = 1. + L*C;
    dummy1 = h1 * S * L * cost * cost / LC1;
    dummy1 -= (sint * LC1);
    dummy1 *= kap;
    dummy1 += (C * cost * LC1 * LC1);
    *Idot = h1 * dummy1;
    *Idot /= (LC1 * LC1 * LC1);
    }
}
```

```
**
**        Program genKA_spect.c
**
**        This program calculates the harmonic power spectrum of the
**        nonlinear signal generated by a hard superconductor when it
**        is driven by an ac magnetic field, with possibly a superposing
**        dc field -- Hd + Ha.cos(wt), according to the generalized
**        critical-state model.  The critical current density
**        is assumed to take the generalized Anderson-Kim form
**
**                    Jc = alpha x c / pow( [|H| + Ho] , beta)
**
**        where alpha = flux-pinning force density when beta = 1,
**              c = speed of light,
**                  Ho = sample dependent parameter.
**
**        In the ungeneralized Kim-Anderson model, beta = 1.
**
**        HR = h(R) = Hd + Ha.cos(wt).
**
**        w = 1.
**
**        Has to be linked with fftcompon.o.
*/

#include <math.h>
#include <stdio.h>
#define MAX 4096  /* 4096 limits N to be <= 12*/
#define N   12    /*No. of data = 2 to the Nth power*/
#define XSPAN1 1.1364    /* size of output graphics */
#define XSPAN2 1.1242    /* size of output graphics */
#define YSPAN  1.94      /* size of output graphics */
#define PI 3.14159265
#define HPI 1.570796327  /* half Pi */
#define QPI 0.785398163  /* quarter Pi */
#define PI2 6.283185307  /* Pi x 2 */
#define ROOT2 1.414213562
#define G(NN,x)             ( A[NN] + B[NN] * x )
#define Br2A(NN,x)          ( B[NN] * x - beta1 * A[NN] )

double pHdHa,mHdHa;  /* pHdHa = Hd + Ha;  mHdHa = Hd - Ha */
double A[11],B[11];  /* A[0] and B[0] not used */
double pi8al,pi4al,inv_pi8al,inv_pi4al,pi4alRbeta1;
double H1star,Ho,Hd,Ha,R;
double beta,beta1,inv_beta1,beta21;

main(argc,argv)

int argc;
char *argv[];

{
   extern double B[11],pi8al,pi4al,inv_pi8al,inv_pi4al;
   extern double H1star,Ho,Ha,Hd,R,mHdHa,pHdHa,pi4alRbeta1;
   extern double beta,beta1,inv_beta1,beta21;

   double cycles,time[MAX],signal[MAX],*ptr_rl[MAX],*ptr_im[MAX],fft_comp[MAX][2];
   double half1_tseries(),half2_tseries(),remain();
   double tmin,tmax,wmin,wmax,tinterv,winterv,t,*freq;
```

```
double ymin,ymax,xmin,xmax,ytic,xtic,posx,posy;
double *welch,inv_Wss,dummyimag = 0.,sqrt_inv_Wss,pi16al,alpha;
char string[50];
int no_data,nodata1,no_data2,no_data21,tt,kk,plotter,labelornot;

/* Hd = dc magnetic field; Ha = ac field amplitude;
  Ho = free parameter;   H* = penetration field. */
 if (argc < 7)
    {
    printf("Usage\: Command Ho Hd Ha H* R beta\n");
    exit(9);
    }

 sscanf(argv[1],"%lf",&Ho);
 sscanf(argv[2],"%lf",&Hd);
 sscanf(argv[3],"%lf",&Ha);
 sscanf(argv[4],"%lf",&H1star);
 sscanf(argv[5],"%lf",&R);
 sscanf(argv[6],"%lf",&beta);

 if (Hd < 0.)
        {
        printf("Hdc must be positive.\n");
        exit(7);
        }

 beta1 = beta + 1.;        inv_beta1 = 1./ beta1;
 beta21 = ( beta + 2.) / beta1;

 tmin = 0.;
 wmin = 0.;     wmax = 102.4;  /*Nyquist freq.*/
 no_data = int_pow(2,N);
 nodata1 = no_data - 1;
 no_data2 = no_data / 2;   no_data21 = no_data2 + 1;
 tinterv = PI / wmax;
 tmax = tinterv * (double)nodata1;
 winterv = 2.*PI/((double)no_data*tinterv);

 freq = (double *)calloc(no_data21,sizeof(double));

 for (tt = 0; tt < no_data; ++tt)
    {
    ptr_rl[tt] = signal+tt;
    ptr_im[tt] = &dummyimag;
    if (tt <= no_data2)
       *(freq+tt) = winterv * (double)tt;
    }

 /* Hanning window, Numerical Recipes pp.425 */
 inv_Wss = 0.;
 welch = (double *)calloc(no_data,sizeof(double));
 for (tt = 0; tt < no_data; ++tt)
    {
    *(welch+tt) = ( PI2 * (double)tt ) / ( (double)no_data-1. );
    *(welch+tt) = cos(*(welch+tt));
    *(welch+tt) = 0.5 * ( 1. - *(welch+tt) );

    inv_Wss += (*(welch+tt) * (*(welch+tt)));
    }
```

```
inv_Wss *= (double)no_data;
inv_Wss = 1./inv_Wss;
sqrt_inv_Wss = sqrt(inv_Wss);

alpha = H1star + Ho;        alpha = pow(alpha,beta1);
alpha -= pow(Ho,beta1);     alpha /= ( 4.* beta1 * PI * R );

pi8al = PI * 8. * alpha;    pi4al = 0.5 * pi8al;
pi4alRbeta1 = pi4al * R * beta1;
inv_pi8al = 1./ pi8al;                  inv_pi4al = 1./ pi4al;

B[10] = B[9] = B[7] = B[6] = B[4] = B[1] = -pi4al * beta1;
B[8] = B[5] = B[3] = B[2] = -B[1];

pHdHa = Hd + Ha;            mHdHa = Hd - Ha;

for (tt = 0; tt < no_data; ++tt)
        {
        if (tt == 0)
          time[0] = tmin;
        else
          time[tt] = time[tt-1] + tinterv;

        if (time[tt] < PI2)             /* first cycle */
          {
          t = time[tt];
          if (time[tt] < PI)            /* first HALF of cycle */
                {
                signal[tt] = half1_tseries(t);
                }
          else                          /* second HALF of cycle */
                {
                signal[tt] = half2_tseries(t);
                }
          }
        else                    /* Beyond the first full cycle */
          {
          t = remain(time[tt],PI2);  /*fold back to 1st full cycle*/
          if (t < PI)                /* first HALF of cycle */
                {
                signal[tt] = half1_tseries(t);
                }
          else                          /* second HALF of cycle */
                {
                signal[tt] = half2_tseries(t);
                }
          }
        }

  for (tt = 0; tt < no_data; ++tt)      *ptr_rl[tt] *= (*(welch+tt));

  /* fft_comp contains real & imag. components for both
          positive and negative frequencies */
  fftcompon(ptr_rl,ptr_im,N,fft_comp);

  printf("xmin, xmax, ymin, ymax, xtic, ytic?\n");
  scanf("%lf %lf %lf %lf %lf %lf",&xmin,&xmax,&ymin,&ymax,&xtic,&ytic);
```

```
/* Graphics preparation */
printf("Which plotter?  1. HP7475A ;  2. HP7470A \: ");
scanf("%d",&plotter);
initgraph("/dev/ttya");
if (plotter == 1)                          /* Set hardcopy output size. */
   window(0.16571,0.18524,XSPAN1+0.16571,YSPAN+0.18524);
else
   window(.08,0.0,XSPAN2+.08,YSPAN);
color(1);                                  /* Choose pen of plotter. */

printf("1. Label;   2.  Do not label \: ");
scanf("%d",&labelornot);

clear();
scale(xmin,xmax,ymin,ymax);          /* Set user's scale w.r.t. plotter's coordinates. */
axes(xmin,ymin,xtic,ytic,1,1);                  /* Draw axes and tick marks. */
axes(xmax,ymax,xtic,ytic,1,1);
border();                                       /* Draw border. */

sprintf(string,"Ho=%5.2lf, Ha=%6.2lf, alpha=%5.2lf, R=%5.2lf"
   ,Ho,Ha,alpha,R);
posx = xmin + (xmax - wmin) * 0.05;
posy = ymin + (ymax - ymin) * 0.95;
if (labelornot == 1)
   {
   move(posx,posy);       label(string);
   }

/* See Numerical Recipes  Fig.12.2.2 for ranges of ii below.
     See also eqt (12.0.14) of the same book.*/
for (tt=0; tt <= no_data/2; ++tt)
   {
   signal[tt] = fft_comp[tt][0] * fft_comp[tt][0];
   signal[tt] += (fft_comp[tt][1] * fft_comp[tt][1]);

   if (tt != 0  &&  tt != (no_data/2))     signal[tt] *= 2.;

   signal[tt] *= inv_Wss;
   signal[tt, = 10. * log10(signal[tt]);

   if (signal[tt] < ymin)     signal[tt] = ymin;

   move(*(freq+tt),signal[tt]);
   if (tt==0)    pendown();
   }
  penup();
  exitgraph();
}


double half1_tseries(t)   /* first half-cycle: 0 < t < PI */

double t;
{
   extern double A[11],B[11],pHdHa,beta1;
   extern double Ho,Hd,Ha,R,pi8al,pi4al,pi4alRbeta1;
   double half1_p_sig(),half1_m_sig(),signal;
   double pHoHR,mHoHR,sinwt,coswt,HR,HRdot,Hacoswt;
```

```
  coswt = cos(t);   sinwt = sin(t);
  Hacoswt = Ha * coswt;
  HR = Hd + Hacoswt;                    HRdot = -Ha * sinwt;
  pHoHR = Ho + HR;   mHoHR = Ho - HR;

  A[1] = pow(pHoHR,beta1) + pi4alRbeta1;
  A[2] = (Ho + pHdHa);
  A[2] = pow(A[2],beta1) - pi4alRbeta1;
  A[3] = pow(mHoHR,beta1) - pi4alRbeta1;
  A[4] = 2.* pow(Ho,beta1) - pow(mHoHR,beta1) + pi4alRbeta1;

  if (HR >= 0.)
        signal = half1_p_sig(HR,HRdot,pHoHR,mHoHR);
  else
        signal = half1_m_sig(HR,HRdot,pHoHR,mHoHR);

  return(signal);
}


double half2_tseries(t)   /* second half-cycle: PI < t < 2.PI */

double t;
{
   extern double pHdHa,mHdHa,pi8al,pi4al,pi4alRbeta1,beta1;
   extern double A[11],B[11],H1star,Ho,Hd,Ha,R;
   double half2_m(),half2_p(),half2_p_II(),signal;
   double half2_III(),coswt,sinwt,pHoHR,mHoHR,Hacoswt,HRdot,HR;

    coswt = cos(t);   sinwt = sin(t);
    Hacoswt = Ha * coswt;
    HR = Hd + Hacoswt;                  HRdot = -Ha * sinwt;
    pHoHR = Ho + HR;          mHoHR = Ho - HR;

    A[5] = (Ho - mHdHa);
    A[5] = pow(A[5],beta1) - pi4alRbeta1;
    A[6] = Ho - mHdHa;                  A[6] = pow(A[6],beta1);
    A[6] = 2.* pow(Ho,beta1) - A[6];
    A[6] += pi4alRbeta1;
    A[7] = pow(mHoHR,beta1) + pi4alRbeta1;
    A[8] = pow(pHoHR,beta1) - pi4alRbeta1;
    A[9] = 2.* pow(Ho,beta1) - pow(pHoHR,beta1) + pi4alRbeta1;
    A[10] = (Ho + mHdHa);               A[10] = pow(A[10],beta1) + pi4alRbeta1;

    if ( Hd <= (Ha - H1star) )
          {
          if (HR <= 0.)
             signal = half2_m(HR,HRdot,pHoHR,mHoHR);
          else
             signal = half2_p(HR,HRdot,pHoHR,mHoHR);
          }

    else if ( Hd <= Ha )
          {
          if (HR <= 0.)
             signal = half2_m(HR,HRdot,pHoHR,mHoHR);
          else if (HR <= (Ha - Hd))
             signal = half2_p(HR,HRdot,pHoHR,mHoHR);
          else
```

```
            signal = half2_p_II(HR,HRdot,pHoHR,mHoHR);
        }
    else
        signal = half2_III(HR,HRdot,pHoHR,mHoHR);

    return(signal);
}


double remain(u,d)

double u,d;
{
    double remain,quot;

    quot = u / d;
    quot = (int)quot;
    remain = u - (d * quot);
    return(remain);
}


double half1_p_sig(HR,HRdot,pHoHR,mHoHR)  /* first half-cycle,  HR >= 0 */

double HR,HRdot,pHoHR,mHoHR;
{
    extern double Ho,Hd,Ha,A[11],B[11];
    extern double R,inv_pi8al,beta,beta1,inv_beta1,beta21;
    double r1,r2,G1r1,G1R,signal;

    r1 = Ho + pHdHa;            r1 = pow(r1,beta1);
    r1 -= pow(pHoHR,beta1);
    r1 *= (inv_pi8al * inv_beta1);
    r1 = R - r1;

    if (r1 < 0.)                r1 = 0.;

    else if (r1 > 1.0001 * R)
        {
        printf("Error\:  r1 > R in the first quadrant.\n");
        exit(3);
        }
    G1r1 = G(1,r1);             G1r1 = pow(G1r1,inv_beta1);
    G1R = G(1,R);               G1R = pow(G1R,inv_beta1);

    signal = G1R * Br2A(1,R) - G1r1 * Br2A(1,r1);
    signal *= (pow(pHoHR,beta) * HRdot);
    signal /= (B[1] * B[1] * beta21);

    return(signal);
}


double half1_m_sig(HR,HRdot,pHoHR,mHoHR)  /* first half-cycle, HR < 0 */

double HR,HRdot,pHoHR,mHoHR;
{
    extern double Ho,Ha,A[11],B[11],pHdHa;
    extern double R,inv_pi8al,inv_pi4al,beta,beta1,inv_beta1,beta21;
```

```
double r1,r2,dummy1,dummy2,signal;
double G3R,G3r2,G4r2,G4r1;

r1 = Ho + pHdHa;                        r1 = pow(r1,beta1);
r1 += pow(mHoHR,beta1);    r1 -= ( 2.* pow(Ho,beta1) );
r1 *= ( -inv_pi8al * inv_beta1 );
r1 += R;
if (r1 > 0.)
        {
        if (r1 > 1.0001 * R)
          {
          printf("Error\: r1 > R in 2nd quadrant.\n");
          exit(4);
          }
        }
else            r1 = 0.;

r2 = pow(mHoHR,beta1) - pow(Ho,beta1);
r2 *= (-inv_pi4al * inv_beta1);
r2 += R;
if (r2 < 0.)                    r2 = 0.;

if (r1 > r2 || r2 > 1.0001 * R)
        {
        printf("Error\: r1 > r2 or r2 > R in 2nd quadrant.\n");
        exit(5);
        }

if (r2 == 0.)                   dummy2 = 0.;
else
        {
        G4r2 = G(4,r2);                 G4r2 = pow(G4r2,inv_beta1);
        G4r1 = G(4,r1);                 G4r1 = pow(G4r1,inv_beta1);
        dummy2 = (Br2A(4,r2) * G4r2 - Br2A(4,r1) * G4r1) / (B[4] * B[4]);
        }

G3R = G(3,R);   G3r2 = G(3,r2);

if (G3R < 0. || G3r2 < 0.)
        {
        printf("Error\: Imaginary number in G3R, or G3r2.\n");
        exit(6);
        }
else
        {
        G3R = pow(G3R,inv_beta1);               G3r2 = pow(G3r2,inv_beta1);
        }

  dummy1 = (Br2A(3,R) * G3R - Br2A(3,r2) * G3r2) / (B[3] * B[3]);
  signal = (dummy1 + dummy2) * pow(mHoHR,beta) * HRdot;
  signal /= beta21;

  return(signal);
}


double half2_m(HR,HRdot,pHoHR,mHoHR)    /* second half-cycle, HR <= 0 */
```

```
double HR,HRdot,pHoHR,mHoHR;
{
  extern double A[11],B[11],Ho,Hd,Ha;
  extern double R,inv_pi8al,beta,beta1,inv_beta1,beta21;
  double r1,r2,G7r1,G7R,signal;

  r1 = Ho - mHdHa;                    r1 = pow(r1,beta1);
  r1 -= pow(mHoHR,beta1);             r1 *= (inv_pi8al * inv_beta1);
  r1 = R - r1;

  if (r1 < 0.)                  r1 = 0.;
  else if (r1 > 1.0001 * R)
        {
        printf("Error\: r1 > R in the first quadrant.\n");
        exit(3);
        }
  G7r1 = G(7,r1);               G7r1 = pow(G7r1,inv_beta1);
  G7R = G(7,R);                 G7R = pow(G7R,inv_beta1);

  signal = G7R * Br2A(7,R) - G7r1 * Br2A(7,r1);
  signal *= ( pow(mHoHR,beta) * HRdot );
  signal /= (B[7] * B[7] * beta21);

  return(signal);
}

/* second half-cycle; for cases:
  (a) if Hd <= (Ha - H1star),  HR > 0;
  (b) if Hd <= Ha,          HR > 0 AND HR <= (Ha - Hd).
*/
double half2_p(HR,HRdot,pHoHR,mHoHR)

double HR,HRdot,pHoHR,mHoHR;

{
  extern double Ho,Ha,A[11],B[11],mHdHa;
  extern double R,inv_pi8al,inv_pi4al,beta,beta1,inv_beta1,beta21;
  double dummy1,dummy2,signal;
  double r1,r2,G8R,G8r2,G9r2,G9r1;

  r1 = Ho - mHdHa;                    r1 = pow(r1,beta1);
  r1 += pow(pHoHR,beta1);             r1 -= (2.* pow(Ho,beta1));
  r1 *= (-inv_pi8al * inv_beta1);
  r1 += R;
  if (r1 > 0.)
        {
        if (r1 > 1.0001 * R)
          {
          printf("Error\: r1 > R in 2nd half-cycle.\n");
          exit(4);
          }
        }
   else                 r1 = 0.;

   r2 = pow(Ho,beta1) - pow(pHoHR,beta1);
   r2 *= (inv_pi4al * inv_beta1);
   r2 += R;
   if (r2 < 0.)                 r2 = 0.;
```

```
 if (r1 > r2 || r2 > 1.0001 * R)
        {
        printf("Error: r1 > r2 or r2 > R in 2nd half-cycle.\n");
        exit(5);
        }

 if (r2 == 0.)                  dummy2 = 0.;
 else
        {
        G9r2 = G(9,r2);                 G9r2 = pow(G9r2,inv_beta1);
        G9r1 = G(9,r1);                 G9r1 = pow(G9r1,inv_beta1);
        dummy2 = (Br2A(9,r2) * G9r2 - Br2A(9,r1) * G9r1) / (B[9] * B[9]);
        }

 G8R = G(8,R);   G8r2 = G(8,r2);

 if (G8R < 0. || G8r2 < 0.)
        {
        printf("Error: Imaginary number in G8R, or G8r2.\n");
        exit(6);
        }
 else
        {
        G8R = pow(G8R,inv_beta1);               G8r2 = pow(G8r2,inv_beta1);
        }

  dummy1 = (Br2A(8,R) * G8R - Br2A(8,r2) * G8r2) / (B[8] * B[8]);
  signal = (dummy1 + dummy2) * HRdot * pow(pHoHR,beta) / beta21;

  return(signal);
}


/* second half-cycle; for case { if Hd <= Ha, HR > (Ha - Hd) } only.
*/
double half2_p_II(HR,HRdot,pHoHR,mHoHR)

double HR,HRdot,pHoHR,mHoHR;
{
   extern double mHdHa,A[11],B[11];
   extern double Ho,Ha,beta,beta1,inv_beta1,beta21;
   extern double R,inv_pi8al,inv_pi4al;
   double r3,G8r3,G8R,signal;

   r3 = Ho - mHdHa;                     r3 = pow(r3,beta1);
   r3 += pow(pHoHR,beta1);              r3 -= ( 2.* pow(Ho,beta1) );
   r3 *= ( inv_pi8al * inv_beta1 );
   r3 = R - r3;

   if (r3 < 0.)                 r3 = 0.;
   else if (r3 > 1.0001 * R)
        {
        printf("Error: r3 > R in the first quadrant.\n");
        exit(3);
        }
   G8r3 = G(8,r3);              G8r3 = pow(G8r3,inv_beta1);
   G8R = G(8,R);                G8R = pow(G8R,inv_beta1);

   signal = G8R * Br2A(8,R) - G8r3 * Br2A(8,r3);
```

```
  signal *= ( HRdot * pow(pHoHR,beta) );
  signal /= (B[8] * B[8] * beta21);

  return(signal);
}


/*  second half-cycle; for case { Hd > Ha } only. */
double half2_III(HR,HRdot,pHoHR,mHoHR)

double HR,HRdot,pHoHR,mHoHR;
{
  extern double mHdHa,A[11],B[11],Ho,Ha;
  extern double beta,beta1,inv_beta1,beta21;
  extern double R,inv_pi8al,inv_pi4al;
  double r1,G8r1,G8R,signal;

  r1 = Ho + mHdHa;                    r1 = pow(r1,beta1);
  r1 = pow pHoHR,beta1) - r1;
  r1 *= ( inv_pi8al * inv_beta1 );
  r1 = R - r1;

  if (r1 < 0.)                  r1 = 0.;
  else if (r1 > 1.0001 * R)
        {
        printf("Error\:  r1 > R in the 2nd half-cycle.\n");
        exit(3);
        }
  G8r1 = G(8,r1);             G8r1 = pow(G8r1,inv_beta1);
  G8R = G(8,R);               G8R = pow(G8R,inv_beta1);

  signal = G8R * Br2A(8,R) - G8r1 * Br2A(8,r1);
  signal *= ( HRdot * pow(pHoHR,beta) );
  signal /= ( B[8] * B[8] * beta21 );

  return(signal);
}
```

```
/*
**        Program  genKA_fftvsHa.c
**
**        This program calculates the different harmonics of the nonlinear
**        signal generated by a type-II superconductor versus ac magnetic
**        field amplitude, according to the generalized critical-state model.
**        The results are stored in an output file.
**
**    The critical current density is assumed to take a generalized
**        Anderson-Kim form
**
**                  Jc = alpha x c / pow( [|H| + Ho] , beta)
**
**        where alpha = flux-pinning force density ( when beta = 1 ),
**              c = speed of light,
**                  Ho = sample dependent parameter,
**                  Hd = dc magnetic field,  Ha = ac field amplitude.
**
**        In the ungeneralized K-A model, beta = 1.
**
**        HR = H(R) = Hd + Ha.cos(wt), w = 1.
**
**        IMPORTANT:  In this printout, the subroutines half1_tseries.c, half2_tseries.c,
**        half1_p_sig.c, half1_m_sig.c, half2_m.c, half2_p.c, half2_p_II.c, and
**        half2_III.c are omitted;  they are listed in the printout of genKA_spect.c.
*/

#include <math.h>
#include <stdio.h>
#define  MAX  2048   /* 2048 limits N to be <=  11*/
#define  N  11      /*No. of data =  2 to the Nth power*/
#define  JUMP  20   /*N = 11 -> JUMP = 20; N = 10 -> JUMP = 10; N = 9 -> JUMP = 5;
              JUMP = w / winterv = 1 / winterv; wmax = 51.2   */
#define  PI  3.14159265
#define  HPI 1.570796327   /* half Pi */
#define  QPI 0.785398163   /* quarter Pi */
#define  PI2 6.283185307   /* Pi x 2 */
#define  ROOT2  1.414213562
#define  N_HARM 40    /* No. of harmonics to be available */
#define  MAX_Ha 500     /* Max. no. of Ha within [Hamin,Hamax]*/
#define  G(NN,x)              ( A[NN] + B[NN] * x )
#define  Br2A(NN,x)           ( B[NN] * x - beta1 * A[NN] )

double pHdHa,mHdHa, A[11],B[11];   /* A[0] and B[0] not used */
double pi8al,pi4al,inv_pi8al,inv_pi4al,pi4alRbeta1;
double H1star,Ho,Hd,Ha,R,beta,beta1,inv_beta1,beta21;

main(argc,argv)

int argc;
char *argv[];

{
   extern double B[11],pi8al,pi4al,inv_pi8al,inv_pi4al;
   extern double H1star,Ho,Ha,Hd,R,mHdHa,pHdHa,pi4alRbeta1;
   extern double beta,beta1,inv_beta1,beta21;

   double cycles,time[MAX],signal[MAX],*ptr_rl[MAX],*ptr_im[MAX];
   double half1_tseries(),half2_tseries(),remain(),fft_comp[MAX][2];
```

```
double tmin,tmax,wmin,wmax,tinterv,winterv,t,*freq;
double pi16al,Hamin,Hamax,Hastep,alpha;
double comp_rl[N_HARM][MAX_Ha],comp_im[N_HARM][MAX_Ha];
double w_har_min[N_HARM],w_har_max[N_HARM],w_harmonic[N_HARM];
double *welch,inv_Wss,dummyimag = 0.,sqrt_inv_Wss;
double coswt,sinwt,pHoHR,mHoHR,Hacoswt,HRdot,HR,LogHa;
char string[50],file[50];
int no_data,nodata1,no_data2,no_data21,no_Ha,hh,kk,tt;
FILE *outfile;

if (argc < 9)
  {
  printf("Usage\: Command Ho Hamin Hamax no_Ha");
       printf(" Hd H* R beta\n");
  exit(9);
  }

sscanf(argv[1],"%lf",&Ho);
sscanf(argv[2],"%lf",&Hamin);
sscanf(argv[3],"%lf",&Hamax);
sscanf(argv[4],"%d",&no_Ha);
sscanf(argv[5],"%lf",&Hd);
sscanf(argv[6],"%lf",&H1star);   /* H1star = penetration field */
sscanf(argv[7],"%lf",&R);                /* Radius of sample. */
sscanf(argv[8],"%lf",&beta);

if (Hd < 0.)
        {
        printf("Hdc must be positive.\n");
        exit(7);
        }

if (no_Ha > MAX_Ha)
        {
        printf("Max number of points for Ha\: %4d.\n",MAX_Ha);
        exit(8);
        }

printf("Enter output file name\: ");
scanf("%s",file);
outfile = fopen(file,"w");

beta1 = beta + 1.;        inv_beta1 = 1. / beta1;
beta21 = ( beta + 2.) / beta1;

tmin = 0.;
wmin = 0.;     wmax = 51.2;  /* Nyquist freq. */
no_data = int_pow(2,N);
nodata1 = no_data - 1;
no_data2 = no_data / 2;   no_data21 = no_data2 + 1;
tinterv = PI / wmax;
tmax = tinterv * (double)nodata1;
winterv = 2. * PI / ((double)no_data * tinterv);

if (Hamax <= 0. || Hamin <= 0.)
   {
   printf("Hamax and Hamin must be positive.\n");
   exit(0);
   }
```

```
else
   {
   Hastep = log10(Hamax);                /* ac H steps in log scale */
   Hastep -= log10(Hamin);
   Hastep /= ((double)no_Ha - 1.);
   }

freq = (double *)calloc(no_data21,sizeof(double));

/*  The following loop prepares for searching the correct peak
   values at various harmonics of the FFT spectrum for each h0.*/
for (kk = 0; kk < N_HARM; ++kk)
   {
   w_harmonic[kk] = (double)(kk+1);

   /* Window to search for harmonics is set below.*/
   w_har_min[kk] = w_harmonic[kk] - winterv * .2;
   w_har_max[kk] = w_harmonic[kk] + winterv * .2;
   }

 for (tt = 0; tt < no_data; ++tt)
   {
   ptr_rl[tt] = signal+tt;                          ptr_im[tt] = &dummyimag;
   if (tt <= no_data2)      *(freq+tt) = winterv * (double)tt;
   }

/* Hanning window, Numerical Recipes pp.425 */
 inv_Wss = 0.;
 welch = (double *)calloc(no_data,sizeof(double));
 for (tt = 0; tt < no_data; ++tt)
   {
   *(welch+tt) = ( PI2 * (double)tt ) / ( (double)no_data-1. );
   *(welch+tt) = cos(*(welch+tt));
   *(welch+tt) = 0.5 * ( 1. - *(welch+tt) );

   inv_Wss += (*(welch+tt) * (*(welch+tt)));
   }
 inv_Wss *= (double)no_data;
 inv_Wss = 1. / inv_Wss;                  sqrt_inv_Wss = sqrt(inv_Wss);

 alpha = H1star + Ho;      alpha = pow(alpha,beta1);
 alpha -= pow(Ho,beta1);    alpha /= ( 4.* beta1 * PI * R );

 pi8al = PI * 8. * alpha;   pi4al = 0.5 * pi8al;
 pi4alRbeta1 = pi4al * R * beta1;
 inv_pi8al = 1. / pi8al;                  inv_pi4al = 1. / pi4al;

 B[10] = B[9] = B[7] = B[6] = B[4] = B[1] = -pi4al * beta1;
 B[8] = B[5] = B[3] = B[2] = -B[1];

 fprintf(outfile,"%10.5lf %9.5lf %10.5lf %8.5lf %3d\n",Ho,Hd,alpha,R,N_HARM);
 fprintf(outfile,"%9.4lf %9.4lf %9.4lf %5.2lf %4d\n"
         ,Hamin,Hamax,H1star,beta,no_Ha);

 LogHa = log10(Hamin);
 for (hh = 0; hh < no_Ha; ++hh)
   {
   Ha = pow(10.,LogHa);
        pHdHa = Hd + Ha;              mHdHa = Hd - Ha;
```

```
    for (tt = 0; tt < no_data; ++tt)
       {
       if (tt == 0)                    time[0] = tmin;
       else                            time[tt] = time[tt-1] + tinterv;

       t = remain(time[tt],PI2);  /*fold back to 1st full cycle*/
       coswt = cos(t);   sinwt = sin(t);
       Hacoswt = Ha * coswt;
       HR = Hd + Hacoswt;                    HRdot = -Ha * sinwt;
       pHoHR = Ho + HR;   mHoHR = Ho - HR;

       if (t < PI)                     /* first HALF of cycle */
              {
              signal[tt] = half1_tseries(HR,HRdot,pHoHR,mHoHR);
              }
       else                            /* second HALF of cycle */
              {
              signal[tt] = half2_tseries(HR,HRdot,pHoHR,mHoHR);
              }
       }

     for (tt = 0; tt < no_data; ++tt)      *ptr_rl[tt] *= (*(welch+tt));

     /* fft_comp contains real & imag. components for both
       positive and negative frequencies */
     fftcompon(ptr_rl,ptr_im,N,fft_comp);

/* See Numerical Recipes  Fig.12.2.2 for ranges of ii below.
  See also eqt (12.0.14) of the same book.*/
for (tt = 0; tt <= no_data2; tt += JUMP)
   {
   for (kk = 0; kk < N_HARM; ++kk)
      {
      /* if within the "window" of the harmonics,...*/
      if ( *(freq+tt) >= w_har_min[kk] && *(freq+tt) <= w_har_max[kk] )
         {
         comp_rl[kk][hh] = fft_comp[tt][0] * sqrt_inv_Wss;
         comp_im[kk][hh] = fft_comp[tt][1] * sqrt_inv_Wss;

         if (tt != 0 && tt != no_data2)
            {
            comp_rl[kk][hh] *= ROOT2;
            comp_im[kk][hh] *= ROOT2;
            }

         break;  /*The right harmonics is found; no need for
            further testing of position of freq */
         }
      }
   }
  LogHa += Hastep;
  }

for (kk = 0; kk < N_HARM; ++kk)
   {
   for (hh = 0; hh < no_Ha; ++hh)
```

```
        {
        fprintf(outfile,"%13.4le ",comp_rl[kk][hh]);
        if (!(hh % 5)) fprintf(outfile,"\n");
        }
      }

  for (kk = 0; kk < N_HARM; ++kk)
      {
      for (hh = 0; hh < no_Ha; ++hh)
        {
        fprintf(outfile,"%13.4le ",comp_im[kk][hh]);
        if (!(hh % 5)) fprintf(outfile,"\n");
        }
      }

  fclose(outfile);                printf("Data stored.\n");
}
```

```
/*
**        Program  genKA_fftvsHd.c
**
**        This program calculates the harmonic power of the nonlinear
**        signal generated by a hard superconductor versus dc magnetic
**        field, according to the generalized critical-state model.
**        The critical current density is assumed to take the
**        generalized Anderson-Kim form
**
**                  Jc = alpha x c / pow( [|H| + Ho] , beta)
**
**        where alpha = flux-pinning force density ( when beta = 1 ),
**              c = speed of light,
**                  Ho = sample dependent parameter.
**          Hd = dc magnetic field,  Ha = ac field amplitud
**
**        In the ungeneralized Kim-Anderson model, beta = 1.
**
**        HR = H(R) = Hd + Ha.cos(wt), w = 1.
**
**    IMPORTANT:  In this printout, the subroutines half1_tseries.c, half2_tseries.c,
**    half1_p_sig.c, half1_m_sig.c, half2_m.c, half2_p.c, half2_p_II.c, and
**    half2_III.c are omitted;  they are listed in the printout of genKA_spect.c.
*/

#include <math.h>
#include <stdio.h>
#define  MAX  2048   /* 2048 limits N to be <= 11 */
#define  N   11      /* No. of data =  2 to the Nth power */
#define  JUMP  20    /* N = 11 -> JUMP = 20; N=10 -> JUMP=10; N=9 -> JUMP=5;
                        JUMP = w / winterv = 1 / winterv; wmax = 51.2   */
#define  PI  3.14159265
#define  HPI 1.570796327   /* half Pi */
#define  QPI 0.785398163   /* quarter Pi */
#define  PI2 6.283185307   /* Pi x 2 */
#define  ROOT2  1.414213562
#define  N_HARM 40         /* No. of harmonics to be available */
#define  MAX_Hd 500        /* Max. no. of Hd within [Hdmin,Hdmax]*/
#define  G(NN,x)             ( A[NN] + B[NN] * x )
#define  Br2A(NN,x)          ( B[NN] * x - beta1 * A[NN] )

double pHdHa,mHdHa,A[11],B[11];    /* A[0] and B[0] not used */
double pi8al,pi4al,inv_pi8al,inv_pi4al,pi4alRbeta1;
double H1star,Ho,Hd,Ha,R,beta,beta1,inv_beta1,beta21;

main(argc,argv)

int argc;
char *argv[];

{
   extern double B[11],pi8al,pi4al,inv_pi8al,inv_pi4al;
   extern double H1star,Ho,Ha,Hd,R,mHdHa,pHdHa,pi4alRbeta1;
   extern double beta,beta1,inv_beta1,beta21;

   double cycles,time[MAX],signal[MAX],*ptr_rl[MAX],*ptr_im[MAX],fft_comp[MAX][2];
   double half1_tseries(),half2_tseries(),remain();
   double tmin,tmax,wmin,wmax,tinterv,winterv,t,*freq;
   double pi16al,Hdmin,Hdmax,Hdstep,alpha;
```

```
double comp_rl[N_HARM][MAX_Hd],comp_im[N_HARM][MAX_Hd];
double w_har_min[N_HARM],w_har_max[N_HARM],w_harmonic[N_HARM];
double *welch,inv_Wss,dummyimag = 0.,sqrt_inv_Wss;
char string[50],file[50];
int no_data,nodata1,no_data2,no_data21,no_Hd,hh,kk,tt;
FILE *outfile;

if (argc < 9)
   {
   printf("Usage\: Command Ho Hdmin Hdmax no_Hd Ha H* R beta\n");
   exit(9);
   }

sscanf(argv[1],"%lf",&Ho);
sscanf(argv[2],"%lf",&Hdmin);
sscanf(argv[3],"%lf",&Hdmax);
sscanf(argv[4],"%d", &no_Hd);
sscanf(argv[5],"%lf",&Ha);
sscanf(argv[6],"%lf",&H1star);   /* H1star = penetration field */
sscanf(argv[7],"%lf",&R);                /* R = radius of the sample */
sscanf(argv[8],"%lf",&beta);

if (Hdmin < 0. || Hdmax < 0.)
        {
        printf("Hdc must be positive.\n");
        exit(7);
        }

if (no_Hd > MAX_Hd)
        {
        printf("Max number of points for Hd\: %4d.\n",MAX_Hd);
        exit(8);
        }

printf("Enter output file name\: ");        scanf("%s",file);
outfile = fopen(file,"w");

beta1 = beta + 1.;        inv_beta1 = 1. / beta1;
beta21 = ( beta + 2.) / beta1;

tmin = 0.;
wmin = 0.;      wmax = 51.2;  /* Nyquist frequency */
no_data = int_pow(2,N);
nodata1 = no_data - 1;
no_data2 = no_data / 2;   no_data21 = no_data2 + 1;
tinterv = PI / wmax;
tmax = tinterv * (double)nodata1;
winterv = 2. * PI / ((double)no_data * tinterv);

Hdstep = (Hdmax - Hdmin) / ((double)no_Hd - 1.);

freq = (double *)calloc(no_data21,sizeof(double));

/*  The following loop prepares for searching the correct peak
    values at various harmonics of the FFT spectrum for each h0. */
for (kk = 0; kk < N_HARM; ++kk)
   {
   w_harmonic[kk] = (double)(kk+1);
   /* Window to search for harmonics is set below.*/
```

```
    w_har_min[kk] = w_harmonic[kk] - winterv * .2;
    w_har_max[kk] = w_harmonic[kk] + winterv * .2;
    }

for (tt = 0; tt < no_data; ++tt)
    {
    ptr_rl[tt] = signal+tt;       ptr_im[tt] = &dummyimag;
    if (tt <= no_data2)           *(freq+tt) = winterv * (double)tt;
    }

/* Hanning window. Numerical Recipes pp.425 */
inv_Wss = 0.;
welch = (double *)calloc(no_data,sizeof(double));
for (tt = 0; tt < no_data; ++tt)
    {
    *(welch+tt) = ( PI2 * (double)tt ) / ( (double)no_data-1. );
    *(welch+tt) = cos(*(welch+tt));
    *(welch+tt) = 0.5 * ( 1. - *(welch+tt) );
    inv_Wss += (*(welch+tt) * (*(welch+tt)));
    }
 inv_Wss *= (double)no_data;
 inv_Wss = 1. / inv_Wss;    sqrt_inv_Wss = sqrt(inv_Wss);

 alpha = H1star + Ho;       alpha = pow(alpha,beta1);
 alpha -= pow(Ho,beta1);    alpha /= ( 4.* beta1 * PI * R );

 pi8al = PI * 8. * alpha;   pi4al = 0.5 * pi8al;
 pi4alRbeta1 = pi4al * R * beta1;
 inv_pi8al = 1./ pi8al;                  inv_pi4al = 1./ pi4al;

 B[10] = B[9] = B[7] = B[6] = B[4] = B[1] = -pi4al * beta1;
 B[8] = B[5] = B[3] = B[2] = -B[1];

 fprintf(outfile,"%10.5lf %9.5lf %10.5lf %8.5lf %3d\n",Ho,Ha,alpha,R,N_HARM);
 fprintf(outfile,"%9.4lf %9.4lf %9.4lf %5.2lf %4d\n"
         ,Hdmin,Hdmax,H1star,beta,no_Hd);

 Hd = Hdmin;
 for (hh = 0; hh < no_Hd; ++hh)
    {
        pHdHa = Hd + Ha;            mHdHa = Hd - Ha;
        for (tt = 0; tt < no_data; ++tt)
           {
           if (tt == 0)             time[0] = tmin;
           else                     time[tt] = time[tt-1] + tinterv;

           if (time[tt] < PI2)      /* first cycle */
                  {
                  t = time[tt];
                  if (time[tt] < PI)          /* first HALF of cycle */
                     {
                     signal[tt] = half1_tseries(t);
                     }
                  else                        /* second HALF of cycle */
                     {
                     signal[tt] = half2_tseries(t);
                     }
                  }
           else                               /* Beyond the first full cycle */
```

```
                {
                t = remain(time[tt],PI2);   /*fold back to 1st full cycle*/
                if (t < PI)                         /* first HALF of cycle */
                   {
                   signal[tt] = half1_tseries(t);
                   }
                else                                /* second HALF of cycle */
                   {
                   signal[tt] = half2_tseries(t);
                   }
                }
          }

       for (tt = 0; tt < no_data; ++tt)      *ptr_rl[tt] *= (*(welch+tt));

       /* fft_comp contains real & imag. components for both positive and
          negative frequencies; see comments on fftcompon.c printout. */
       fftcompon(ptr_rl,ptr_im,N,fft_comp);

 /* See Numerical Recipes  Fig.12.2.2 for ranges of ii below.
    See also eqt (12.0.14) of the same book.*/

 for (tt = 0; tt <= no_data2; tt += JUMP)
    {
    for (kk = 0; kk < N_HARM; ++kk)
       {
       /* if within the "window" of the harmonics,...*/
       if ( *(freq+tt) >= w_har_min[kk] && *(freq+tt) <= w_har_max[kk] )
          {
          comp_rl[kk][hh] = fft_comp[tt][0] * sqrt_inv_Wss;
          comp_im[kk][hh] = fft_comp[tt][1] * sqrt_inv_Wss;

          if (tt != 0 && tt != no_data2)
             {
             comp_rl[kk][hh] *= ROOT2;
             comp_im[kk][hh] *= ROOT2;
             }

          break;  /*The right harmonics is found; no need for
                       further testing of position of freq */
          }
       }
    }
  Hd += Hdstep;
  }

for (kk = 0; kk < N_HARM; ++kk)
   {
   for (hh = 0; hh < no_Hd; ++hh)
      {
      fprintf(outfile,"%13.4le ",comp_rl[kk][hh]);
      if (!(hh % 5)) fprintf(outfile,"\n");
      }
   }

for (kk = 0; kk < N_HARM; ++kk)
   {
```

```
    for (hh = 0; hh < no_Hd; ++hh)
        {
        fprintf(outfile,"%13.4le ",comp_im[kk][hh]);
        if (!(hh % 5)) fprintf(outfile,"\n");
        }

fclose(outfile);
printf("Data stored.\n");
```

```
**
**      Routine odeint1.c
**      This is modified from ODEINT() of Numerical Recipes,
**      (by Press, Flannery, Teukolsky and Vetterling, Cambridge
**      University Press, 1986), pp.559, a Runge-Kutta driver
**      with adaptive stepsize control. Integrate the 'nvar'
**      starting values 'ystart' from 'x1' to 'x2' with accuracy
**      'eps'. 'dh1' should be set as a guessed first stepsize.
**      'ystart' is REPLACED by values at the end of the integration
**      interval. (*derivs)() is the user-supplied subroutine
**      for calculating the right-hand side derivative. 'slope1'
**      will store the rhs derivatives at the STARTING point 'x1',
**      (input)'ystart'.
**
**      x2 can be smaller than x1; in that case, the steps are
**      AUTOMATICALLY made in the negative direction.
**
**      Has to be linked to rkqcrk4.c. This program was originally
**      written on SUN 3/50, Unix.
*/

#include <math.h>
#define MAXSTEP  5000
#define MAXSTEP1 4999   /* = MAXSTEP - 1 */
#define NMAX  10        /* max no. of eqns in the set */
#define TWO  2.0
#define ZERO  0.0
#define TINY  1.e-30

odeint1(ystart,nvar,x1,x2,eps,dh1,derivs,slope1)

double *ystart,x1,x2,eps,dh1,*slope1;
int nvar,(*derivs)();

{
   double yscal[NMAX],y[NMAX],dydx[NMAX],x,*x_ptr,dh;
   double dhdydx,dhdid,dhnext,*dhdidptr,*dhnextptr;
   register int nstp,ii;

   x_ptr = &x;          dhdidptr = &dhdid;
   dhnextptr = &dhnext;

   x = x1;
   dh = fabs(dh1) ;
   if (x2 < x1)   dh *= -1.;

   for (ii=0; ii<nvar; ++ii)
          y[ii] = *(ystart+ii);

   for (nstp = 0; nstp < MAXSTEP; ++nstp)  /*Take at most MAXSTEP steps*/
          {
          (*derivs)(x,y,dydx);
          if (nstp == 0)
             {
             for (ii = 0; ii < nvar; ++ii)
                    *(slope1+ii) = dydx[ii];
             }
```

```
        /* Scaling used to monitor accuracy. This general-
          purpose choice can be modified if need be. */
        for (ii = 0; ii < nvar; ++ii)
          {
          /*   yscal[ii] = 1.;  */
          dhdydx = dh * dydx[ii];
          if (dhdydx < 0.)   dhdydx *= -1.;  /*absolute value*/
          yscal[ii] = fabs(y[ii]) + dhdydx;

           yscal[ii] += TINY;
           }

        /* If step can overshoot end. cut down stepsize.*/
        if ( ((x+dh-x2) * (x+dh-x1)) > 0.)
          dh = x2 - x;

        /* x,*y will be replaced by new values. */
        rkqc(y,dydx,nvar,x_ptr,dh,eps,yscal,dhdidptr,dhnextptr,derivs);

        if ( ((x-x2) * (x2-x1)) >= 0.)  /* Are we done? */
          {
          for (ii = 0; ii < nvar; ++ii)
                *(ystart+ii) = y[ii];   /* results */
          break;   /*Normal exit*/
          }
        else
          {
          dh = dhnext;
          if (nstp == MAXSTEP1)
                printf("Too many steps.\n");
          }
         }
}
```

```
/*
**        Routine rkqcrk4.c.
**        This is a C version of RKQC() of Numerical Recipes
**        (by Press, Flannery, Teukolsky and Vetterling, Cambridge
**        University Press, 1986) pp.558.   It is a stepper program
**        that takes one "quality-controlled" Runge-Kutta step.
**
**    From Num. Rec.:  5th-order Runge-Kutta step with monitoring
**        of local truncation error to ensure accuracy and
**        adjust stepsize. Input are the dependent variable vector
**        'y' of length 'n' and its derivative 'dydx' at the starting
**        value of the independent variable 'x'.  Also input are the
**        stepsize to be attempted 'dhtry', the required accuracy
**        'eps', the vector 'yscal' against which the error is
**        scaled, and any parameters needed for the particular
**        function involved.  On output, *y and *x_ptr are REPLACED
**        by their new values, 'dhdid' is the stepsize which was
**        actually accomplished, and 'dhnext' is the estimated next
**        stepsize.  'derivs()' is the user-supplied subroutine that
**        computes the right-hand side derivatives ( a function
**        pointer).
*/

#include <math.h>
#define NMAX  10
#define PGROW  -0.20
#define PSHRNK  -0.25
#define FCOR  0.066666666666667  /* = 1./15. */
#define ONE  1.
#define SAFETY  0.9
#define ERRCON  6.e-4                    /* = pow(4 / SAFETY, 1 / PGROW) */

rkqc(y,dydx,n,x_ptr,dhtry,eps,yscal,dhdidptr,dhnextptr,derivs)

double *y,*dydx,*x_ptr,dhtry,eps,*yscal,*dhdidptr,*dhnextptr;
int n, (*derivs)();

{
   double ytemp[NMAX],ysav[NMAX],dysav[NMAX],xsav,dh;
   double dhh, errmax,compare;
   int ii,repeat;

   xsav = *x_ptr;     /*Save initial values.*/
   for (ii = 0; ii < n; ++ii)
           {
           ysav[ii] = *(y+ii);
           dysav[ii] = *(dydx+ii);
           }

   dh = dhtry;    /* Set stepsize to the initial trial value*/

   repeat = 1;
   while (repeat == 1)
           {
           dhh = 0.5 * dh;   /* Take two half steps */
           nrrk4(ysav,dysav,n,xsav,dhh,ytemp,derivs);
           *x_ptr = xsav + dhh;
           (*derivs)(*x_ptr,ytemp,dydx);
           nrrk4(ytemp,dydx,n,*x_ptr,dhh,y,derivs);
```

```
        *x_ptr = xsav + dh;
        if (*x_ptr == xsav)
          {
          printf("Stepsize not significant in rkqc().\n");
          exit(0);
          }
        nrrk4(ysav,dysav,n,xsav,dh,ytemp,derivs);   /* Take the large step */

        errmax = 0.;
        for (ii = 0; ii < n; ++ii)
          {
          ytemp[ii] -= *(y+ii);    /* error estimate */
          compare = ytemp[ii] / (*(yscal+ii));
          if (compare < 0.)  compare *= -1.;   /* absolute value */
          if (compare > errmax)
                errmax = compare;
          }
        errmax /= eps;  /*Scale relative to required tolerance*/

        if (errmax > ONE)  /* Truncation error too large, reduce stepsize */
          {
          dh *= (SAFETY * pow(errmax,PSHRNK));
          repeat = 1;   /* For another try */
          }
        else   /* Step succeeded. Compute size of next step.*/
          {
          repeat = 0;
          *dhdidptr = dh;
          if (errmax > ERRCON)
                *dhnextptr = SAFETY * dh * pow(errmax,PGROW);
          else
                *dhnextptr = 4. * dh;
          }
        }

  for (ii = 0; ii < n; ++ii)    /* Mop up 5th order truncation error.*/
        *(y+ii) += (ytemp[ii] * FCOR);

}


/*
**      Routine nrrk4.c
**      This is a C version of RK4() of Numerical Recipes,
**      pp.553.
**    From N.R.: Given values for 'n' variables 'y' and
**      their derivatives 'dydx' known at 'x', use the 4th
**      order Runge-Kutta method to advance the solution
**      over an interval 'dh' and return the incremented
**      variables as 'yout', which need not be a distinct
**      array from 'y'. The user supplies the subroutine
**      (*derivs)(x,y,dydx) which returns derivatives
**      'dydx' at 'x'.
*/

#include <math.h>
#define NMAX  10
```

```
nrrk4(y,dydx,n,x,dh,yout,derivs) /*yout is the only output*/

double *y,*dydx,*yout;
double x,dh;
int n,(*derivs)();

{
   double dhh,dh6,xh,yt[NMAX],dyt[NMAX],dym[NMAX],xph;
   register int ii;

   dhh = dh * 0.5;
   dh6 = dh * 0.166666666666667;  /* dh / 6. */
   xh = x + dhh;
   for (ii = 0; ii < n; ++ii)    /* 1st step */
            {
            yt[ii] = *(y+ii) + dhh * (*(dydx+ii));
            }
    (*derivs)(xh,yt,dyt);        /* 2nd step */
    for (ii = 0; ii < n; ++ii)
            {
            yt[ii] = *(y+ii) + dhh * dyt[ii];
            }
    (*derivs)(xh,yt,dym);        /* 3rd step */
    for (ii = 0; ii < n; ++ii)
            {
            yt[ii] = *(y+ii) + dh * dym[ii];
            dym[ii] += dyt[ii];
            }
    xph = x + dh;
    (*derivs)(xph,yt,dyt);       /* 4th step */
    for (ii = 0; ii < n; ++ii)   /* Accumulate increments with proper weights */
            {
            *(yout+ii) = *(y+ii) + dh6 * ( *(dydx+ii)
                        + dyt[ii] + dym[ii] + dym[ii]);
            }
}
```

```
*
**      Function fftcompon.c
**      This is the basically an FFT program I wrote in 1986
**      (ie. fft1.c in library FFT). It outputs the real and
**      imaginary fourier components of the input data.
**      Reference: "The Fast Fourier Transform," by E. Oran Brigham.
**      Prentice-Hall (1974).
**
**      The data are assumed to be composed of complex numbers.
**      Program checked on Mar 12, 1988.
*/

#include <math.h>
#define PI 3.141592654
#define PI2 6.283185307

** ptr_rl is the pointer to the real part of the input data;
** ptr_im is the pointer to the imaginary part of the input data;
** number of (complex) input data points = 2 to the "gamma"th power.
** x1 is the (2 to the gamma)x(2) array which contains real and imaginary
** components for both positive and negative frequencies.
*/

fftcompon(ptr_rl,ptr_im,gamma,x1)

double *ptr_rl[],*ptr_im[],*x1;
unsigned gamma;

{
   double *w,cplx_dum2[2],cplx_dum3[2],cplx_dum1[2];
   double *x0,argu;
   unsigned samp_no,samp_no2,samp_no22;
   unsigned pindex,kindex,lindex,get_p(),pindex2,kindex2;
   int kk,jj,ii,step,dual_space,ii2;
   int kk_max,dual_space2;
   char *calloc();

   samp_no = int_pow(2,(int)gamma);  /* No. of samples input */
   samp_no22 = samp_no * 2;
   samp_no2 = samp_no / 2;

   x0 = (double *)calloc(samp_no22,sizeof(double));
   w = (double *)calloc(samp_no,sizeof(double));

   if (x0==0 || w==0 || x1==0)
         {
         printf("Warning\:  Not enough free memory.\n");
         /*  exit(0);  */
         }

   for (ii =0; ii<samp_no; ++ii)
         {
         ii2 = ii * 2;
     *(x0 + ii2) = *ptr_rl[ii];
         *(x0 + ii2 + 1) = *ptr_im[ii];
         }
```

```
for (pindex=0; pindex < samp_no2; ++pindex)
        {
        pindex2 = pindex * 2;
        argu = PI2 * (double)pindex / (double)samp_no;
        *(w+pindex2) = cos(argu);
        *(w+pindex2+1) = sin(argu);
        }

for (lindex=1; lindex <= gamma; ++lindex)
   {
   dual_space = int_pow(2,(int)(gamma-lindex));
        kk_max = samp_no2 / dual_space;
        dual_space2 = dual_space * 2;

        step = 0;
        for (kk = 0; kk < kk_max; ++kk)
           {
           for (ii=0; ii<dual_space; ++ii)
                   {
                   kindex = ii + step;
                   kindex2 = kindex * 2;
                   if (kindex > (samp_no - 1))
                      {
                      printf("kindex too large\n");
                      }

                   pindex = get_p(kindex, gamma, lindex);

                   for (jj=0; jj<=1; ++jj)
                     {
                     cplx_dum2[jj] = *(w + 2*(pindex % samp_no2)+jj);
                     cplx_dum3[jj] = *(x0 + kindex2 + dual_space2 + jj);
                     }

                   comp_mult(cplx_dum2,cplx_dum3,cplx_dum1);

                   for (jj=0; jj<=1; ++jj)
                     {
                     *(x1 + kindex2 + jj)
                             = *(x0+kindex2+jj) + cplx_dum1[jj];
                     *(x1 + kindex2 + dual_space2 + jj)
                             = *(x0+kindex2+jj) - cplx_dum1[jj];
                     }
              }
           step += dual_space2;
           }

        for (ii=0; ii < samp_no; ++ii)
           {
           ii2 = ii * 2;
           for (jj=0; jj<=1; ++jj)
              {
              *(x0 + ii2 + jj) = *(x1 + ii2 +jj);
                 }
           }
        }
```

```
  cfree(w);
  unscramble(x0,gamma,x1);
  cfree(x0);    /* added on 9/5/1988 */
}

unscramble(x,gamma,fourier)   /* checked 9/21/86  */

double *x, *fourier;
unsigned gamma;

{
  int ii,kk,ll,num,gamma1,ll2,kk2;

  num = int_pow(2,(int)gamma);
  gamma1 = (int)gamma - 1;

     for (kk=0; kk < num; ++kk)
        {
        ll = 0;
        for (ii=0; ii <= gamma1; ++ii)
           {
           ll = ll | (((kk>>ii) & 1) << (gamma1 -ii));
           }

              kk2 = kk * 2;                ll2 = ll * 2;
        *(fourier + ll2) = *(x + kk2);
        *(fourier + ll2 + 1) = *(x + kk2 + 1);
        }
}

unsigned get_p(k,gamma,l)

unsigned k, gamma, l;

{
   unsigned k1, p, kdum;
   int ii;

   k1 = k >> (gamma - l);

   p = 0;

   for (ii=0; ii <= (gamma - 1); ++ii)
      {
      kdum = ((k1 >> ii) & 1) << (gamma - 1 - ii);
      p = p | kdum;
      }

   return(p);
}

comp_mult(z1,z2,z1z2)                /* Multiply complex nos. z1 and z2 to become z1z2. */

double z1[],z2[],z1z2[];

{
   z1z2[0] = z1[0] * z2[0] - z1[1] * z2[1];
   z1z2[1] = z1[0] * z2[1] + z1[1] * z2[0];
}
```

```
int_pow(x,n)            /*  x to the nth power  */

int x,n;

{
   int ii, ans = 1;

   for (ii=1; ii<=n; ++ii)
      {
            ans *= x;
            }
   return(ans);
}
```

Appendix C GPIB Operation Details — Programs for Data Acquisition

In this appendix, we provide the computer programs which the author has written and actually used to take and plot some of the data presented in Chapter 4.

A list of the programs used for experimental data acquisition is given in Table C.1. Programs for plotting out the data taken by the programs in Table C.1 are listed in Table C.2. In Table C.3, some general purpose routines that are written to control some of the electronic apparatus used in the experiments, or to make graphics on the video and the plotter, and needed for the some of the programs listed in Tables C.1 and C.2, are also listed. All these programs are written in Turbo C and are given in their complete form later in the appendix.

Table C.1 Programs for experiment data acquisition.

Program name	Purpose(s)
sadump.c	Take the spectrum and all other information currently on the screen of the HP 3585A spectrum analyzer and store the data in a file.
pnf_h1.c	Take specified harmonic power P(nf) as a function of ac field amplitude and store the data in a file.
xpxpph1n.c	Take the inductive and dissipative components of the ac susceptivity as a function of ac field amplitude and store the data in a file. The superposing dc magnetic field is constantly monitored by a DMM.
xpxpph1t.c	Same as xpxpph1n.c, except that thermocouple voltage, instead of dc magnetic field, is being monitored.

Table C.2 Graphics programs for outputting experiment data.

Program name	Purpose(s)
pltspc.c	Reads a data file written by sadump.c and output the data to HP 7225B.
plotpnf.c	Reads a data file written by pnf_h1.c and output the data to the video monitor.
hplotpnf.c	Same as plotpnf.c, except that the data are output to HP 7225B.
vplotf1.c	Reads a data file written by xpxpph1n.c and output the data to the video monitor.

Table C.3 General purpose routines used for electronics and graphics.

Program name	Purpose(s)
keithley.c	Routines for controlling Keithley 197 DMMs through National Instrument GPIB-PCII.
par5209.c	Routines for controlling PARC 5209 lock-in amplifiers through National Instrument GPIB-PCII.
hp3325.c	Routines for controlling HP 3325A synthesizers through National Instrument GPIB-PCII.
hp3585sa.c	Routines for controlling HP 3585A spectrum analyzers through National Instrument GPIB-PCII.
hpgraph.c	Graphics routines for HP 7225B plotter, through National Instrument GPIB-PCII.
v_graph.c	Turbo C graphics routines for the video monitor.

```
/*
**   Program:    sadump.c
**   This program asks HP3585A spectrum analyzer to dump
**   all current information on the analyzer's screen to
**   the computer thru the National Instrument GPIB-PCII.
**
**   This program has to be linked with \gpib-pc\tc\tcibs.obj.
*/

#include <stdio.h>
#include "decl.h"
#define LENGTH 12012

main()

{
    FILE *file;
    char string[100],rd[28],lngstg[LENGTH];
    int ud,ii;

    printf("Enter output filename:  ");
    scanf("%s",string);

    file = fopen(string,"w");

    printf("Manually set the conditions of SA1; hit <CR> when ready.");
    getchar();
    getchar();

    ud = ibfind("sa1");

    ibwrt(ud,"D7",2);
    ibwrt(ud,"T4",2);

    ibrd(ud,rd,16);
    for (ii=0; ii<16; ++ii)
        fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,27);
    for (ii=0; ii<27; ++ii)
        fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,11);
    for (ii=0; ii<11; ++ii)
        fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,17);
    for (ii=0; ii<17; ++ii)
        fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,16);
    for (ii=0; ii<16; ++ii)
        fprintf(file,"%c",rd[ii]);
```

```
    ibrd(ud,rd,24);
    for (ii=0; ii<24; ++ii)
          fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,22);
    for (ii=0; ii<22; ++ii)
          fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,13);
    for (ii=0; ii<13; ++ii)
          fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,13);
    for (ii=0; ii<13; ++ii)
          fprintf(file,"%c",rd[ii]);

    ibrd(ud,rd,14);
    for (ii=0; ii<14; ++ii)
          fprintf(file,"%c",rd[ii]);

    for (ii=0; ii<LENGTH; ++ii)
          lngstg[ii] = ' ';

    ibwrt(ud,"D3",2);
    ibwrt(ud,"T4",2);

    ibrd(ud,lngstg,LENGTH);
    for (ii=0; ii<LENGTH; ++ii)
          {
          fprintf(file,"%c",lngstg[ii]);
          if ( (ii%60) == 0 )
              fprintf(file,"\n");
          }

    fclose(file);   ibloc(ud);
    printf("Use vpltspc (video) or pltspc (plotter) to plot the results.\n");
}
```

```
/*
**      Pnf_H1.c:  This program is designed to take data
**      for experiments measuring harmonic power P(nf) (of
**      high-Tc superconductors) vs. ac magnetic field H1.
**      H1 field is stepped in logarithmic scale.
**
**      This program is configured by Harry Lam in August, 1990;
**      it has to be linked with hp3585sa.c, hp3325.c, keithley.c,
**      and \gpib-pc\tc\tcibs.obj.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include "decl.h"
#define REAL  float

int board_ud;

main()
   {
   int udsyn1,udsa1,udvmdc,udvmac;
   int yesno,rngindex,harm_no,voltam;
   int samp_no,ii,jj,vv,AVE_NO,coil,mask,SAMP14;
   int sa_alnum(),sasetrng(),sa_manF();    /* Routines for spec. anlyzer */

   REAL synparam(),synphase(),synampl();   /* Routines for synthesizers */
   REAL synfreq();
   REAL sa_mkamp(),sa_Rconv();             /* Routines for lock-in */
   REAL keith_rd();                        /* Routines for Keithley DMM */
   REAL rnglvl,P1f,Pnf,fundfreq,harmfreq;
   REAL ac_monR,ac_calib,ac_monV,H1;
   REAL dc_monR,dc_calib,dc_monV,Hdc;
   REAL syn_ph,offset,syn_amp,syn_f,ref_min,drv_max;
   REAL max_H1,amplit,amplit2,log_amp,logstep;

   char ref[15],mkrfrq[26],dbdiv[10],range[16],mkramp[15];
   char ctrfrq[23],span[21],rbw[12],vbw[12],st[13];
   char osunit[5],func[10],hivolt[11],synunit[6],synfunit[4];
   char Vokay,Vunit[4],dummy[5],lastpt[10],temp[10];
   char filename[20],sample[20],date[20],receiver[30];
   char overwrt[5],append,comment[80],c_yesno[5],term_yn[5];
   char NAD_yn[5],dc_yesno[5];
   FILE *outfile;

   printf("\007\n\nEnter output filename:  ");
   scanf("%s",filename);

   printf("\007\n\New file or OVERWRITE ? \(\"yes\" or \"no\"\)  ");
   scanf("%s",overwrt);
   if (overwrt[0] == 'y' && overwrt[1] == 'e' && overwrt[2] == 's')
        {
        outfile = fopen(filename,"w");              append = 'n';
        }
```

```
else
      {
      outfile = fopen(filename,"a");              append = 'y';
      printf("\007\n** Appending to previously existing file. **\n");
      }

if (outfile == NULL)
      {
      printf("File cannot be open.\n");
      exit(0);
      }

if (append == 'n')
      {
      printf("\007\nEnter sample number (20 characters max):  ");
      scanf("%s",sample);

      printf("\007\nEnter today's date (mon-day-year):  ");
      scanf("%s",date);

      printf("\007\nEnter sample temperature (Kelvin):  ");
      scanf("%s",temp);

      printf("\007\n(1) One-coil receiver? (2) Two-coil receiver?   ");
      scanf("%d",&coil);
      if (coil == 1)
         sprintf(receiver,"One-coil receiver");
      else
         sprintf(receiver,"Two-coil receiver");
      }

printf("\007\nMake sure that synth.1, spec. analyzer and vm2 are on.");
printf("  Hit <CR> when ready.\n");
getchar();              getchar();

printf("\007\nWill vm1 be used to monitor Hdc? ('y' or 'n')  ");
scanf("%s",dc_yesno);
if (dc_yesno[0] == 'y')
      {
      udvmdc = keith_in("vm1");
      printf("\007\nMake sure vm1 is on and set correctly.\n");
      printf("\007\nEnter the value");
      printf(" of the dc monitoring resistance in Ohms:  ");
      scanf("%f",&dc_monR);

      printf("\007\nEnter calibration of dc-coil in H(G) / I(A):  ");
      scanf("%f",&dc_calib);
      }
else
      {
      printf("\007\nEnter superposing dc magnetic field (Gauss):  ");
      scanf("%f",&Hdc);
      }
```

```
getchar();
board_ud = ibfind("gpib0");
udsyn1 = ibfind("syn1");
udsa1 = ibfind("sa1");      udvmac = keith_in("vm2");
ibsic(board_ud);

mask = ERR | TIMO | SRQI | RQS | CMPL;
ibtrap(mask,2);         /* 1: off; 2: record but no trap; 3: both record
                           and trap */

printf("\007\nEnter harmonic number wanted:  ");
scanf("%d",&harm_no);

printf("\007\nWill NAD amplifier be used? ('y' or 'n') ");
scanf("%s",NAD_yn);
if (NAD_yn[0] == 'y')
     {
     printf("\007\nIs syn1's ");
     printf("output terminated by 50 Ohms? ('y' or 'n') ");
     scanf("%s",term_yn);
     if (term_yn[0] == 'y')    drv_max = 4.;
     else                      drv_max = 2.;
     }
else
     {
     printf("\007\nEnter max. syn1's Vpp:  ");
     scanf("%f",&drv_max);
     }

printf("\007");
printf("\nComputer's ready to read gen'l settings of synthesizer 1;\n");
printf("hit <CR> if synthesizer 1 is set.\n");
getchar();              getchar();

offset = synparam(udsyn1,osunit,func,hivolt);
syn_f = synfreq(udsyn1,synfunit);

/* Convert syn_f into kHz for fundfreq, if needed */
switch(synfunit[0])
     {
     case 'H':   fundfreq = syn_f / 1000.;   break;
     case 'k':   fundfreq = syn_f;               break;
     case 'M':   fundfreq = syn_f * 1000.;
     }
harmfreq = fundfreq * (REAL)harm_no;

printf("\007Computer's ready to read gen'l settings of HP3385A; ");
printf("hit <CR> if it is set.");
getchar();

rngindex
= sa_alnum(udsa1,ref,mkrfrq,dbdiv,range,mkramp,ctrfrq,span,rbw,vbw,st);

rnglvl = sa_Rconv(rngindex);
```

```
printf("\007\nIs ac current monitored by vm2 as voltmeter \(1\)");
printf(" or ammeter \(2\): ");
scanf("%d",&voltam);
if (voltam == 1)
      {
      printf("\007\nEnter the value of ");
      printf("the ac monitoring resistance in Ohms: ");
      scanf("%f",&ac_monR);
      }

printf("\007\nEnter calibration of ac-coil in H1(G) / Irms(A): ");
scanf("%f",&ac_calib);

ibloc(udsyn1);
printf("\nSynthesizer 1's settings:\n");
printf("offset = %6.2f %s, %s, %s, %11.4f %s\n"
            ,offset,osunit,func,hivolt,syn_f,synfunit);
printf("\nInitial spec. analyzer settings:\n");
printf("%s, %s, %s, %s\n",ref,mkrfrq,dbdiv,range);
printf("%s, %s, %s, %s\n",mkramp,ctrfrq,span,st);
printf("%s, %s\n",rbw,vbw);

printf("\007\nHow many points to average? ");
scanf("%d",&AVE_NO);

if (append == 'n')
      {
      printf("\007\n");
      printf("Want to add some comment ( < 80 char, 'y' or 'n')? ");
      scanf("%s",c_yesno);
      if (c_yesno[0] == 'y')
         {
         getchar();
         printf("Comment: "); gets(comment);
         }

      fprintf(outfile,"File %s\n",filename);
      fprintf(outfile,"Date: %s, Sample: %s, Temp: %s K\n"
         ,date,sample,temp);

      if (dc_yesno[0] == 'y')
         {
         fprintf(outfile,"Dc magnetic field monitored.\n");
         fprintf(outfile,"Dc mon. resistance (Ohms): %7.3f\n",dc_monR);
         fprintf(outfile,"Calibration of dc-coil H(G)/I(A): %8.3f\n",
               dc_calib);
         }
      else
         fprintf(outfile,"Dc magnetic field not monitored.\n");

      if (NAD_yn[0] == 'y')
         fprintf(outfile,"NAD amplifier used. ");
      else
         fprintf(outfile,"NAD amplifier not used. ");
```

```
        fprintf(outfile,"%s\n",receiver);
        fprintf(outfile,"\nSynthesizer 1's settings:\n");
        fprintf(outfile,"offset = %6.2f %s, %s, %s, %11.4f %s\n"
                ,offset,osunit,func,hivolt,syn_f,synfunit);
        fprintf(outfile,"\nInitial spec. analyzer settings:\n");
        fprintf(outfile,"%s, %s, %s, %s\n",ref,mkrfrq,dbdiv,range);
        fprintf(outfile,"%s, %s, %s\n",mkramp,ctrfrq,span);
        fprintf(outfile,"%s, %s, %s\n",st,rbw,vbw);

        if (voltam == 1)
           fprintf(outfile,"Ac monitoring resistance in Ohms:  %7.3f\n"
                ,ac_monR);

        fprintf(outfile,
                "Calibration of ac-coil in H1(G) / Irms(A):  %8.3f\n\n"
                ,ac_calib);

        fprintf(outfile,"# of ave. / data point = %d",AVE_NO);
        fprintf(outfile,", harmonic no. = %3d\n",harm_no);

        if (c_yesno[0] == 'y')    fprintf(outfile,"%s\n\n",comment);

        fprintf(outfile,"*****\n");
        if (voltam == 1)
           fprintf(outfile,"  No    ac_monV       syn_amp       Hdc(G)  ");
        else
           fprintf(outfile,"  No    ac_monA       syn_amp       Hdc(G)  ");

        fprintf(outfile,"   P(%2df)(dBm)     H1(G)\n\n",harm_no);
        }

    ibwrt(udsyn1,"AM",2);    ibloc(udsyn1);
    printf("\007Input the starting syn1 amplitude (volts):  ");
    scanf("%f",&amplit);
    logamp = log10(amplit);
    printf("\007Input max. H1 desired:  ");
    scanf("%f",&max_H1);
    printf("\007Input no. of samples per decade:  ");
    scanf("%d",&SAMP14);

    printf("\007\007\007\007\n");
    printf("About to start taking data.\n\n");
    printf("\007\nHit <CR> when ready.");
    getchar();    getchar();

    logstep = 1. / ((REAL)SAMP14);   /* samples per decade */
    samp_no = 1;
    for ( ; ; )
        {
        amplit = pow(10.,logamp);
        if (amplit > drv_max)
            {
            ibloc(udsyn1);
            printf("\007\007\007\nMANUALLY reset NAD amplifier ");
```

```
        printf("input level and synth. 1 amplitude.\n");
        printf("\nEnter new syn1 amplitude ");
        printf("(in volts, or 99999 to quit): ");
        scanf("%f",&amplit);
        if (amplit > drv_max && amplit < = 40.)
            {
            printf("\007\007\007\nEntry");
            printf(" larger than %6.2f volts, reenter to reconfirm: "
                ,drv_max);
            scanf("%f",&amplit);
            }
        logamp = log10(amplit);
        if (amplit > 40.)   break;
        }

    sa_port(udsa1,2);          synsetam(udsyn1,amplit);
    sa_port(udsa1,1);          sleep(1);

    sa_manF(udsa1,fundfreq);
    for (ii=0; ii < 11; ++ii)
        {
        P1f = sa_mkamp(udsa1);

        if (P1f > (rnglvl - 10.) && rngindex < 12)
            {
            ++rngindex;              sasetrng(udsa1,rngindex);
            rnglvl = sa_Rconv(rngindex);
            }
/*      else if (P1f < (rnglvl - 40.) && rngindex > 1)
            {
            --rngindex;              sasetrng(udsa1,rngindex);
            rnglvl = sa_Rconv(rngindex);
            }
*/
        else if (P1f > rnglvl && rngindex = = 12)
            {
            printf("\007\007\007\nWARNING: ");
            printf("Input to spec. analyzer is too large!!!\n");
            printf("Correct it and hit <CR> to continue.\n");
            ibloc(udsa1);
            fclose(outfile);     outfile = fopen(filename,"a");
            getchar();       getchar();       break;
            }
        else
            break;
        }

    sa_manF(udsa1,harmfreq);       sleep(1);

    Pnf = sa_mkamp(udsa1);
    if (Pnf > (rnglvl - 6.* harm_no) && rngindex < 12)
        {
        ++rngindex;
        ++rngindex;          sasetrng(udsa1,rngindex);
```

```
    rnglvl = sa_Rconv(rngindex); sleep(1);
    }

Pnf = 0.;
for (ii=0; ii<AVE_NO; ++ii)    Pnf += sa_mkamp(udsa1);
Pnf /= ((REAL) AVE_NO);

/* Read ac monitor voltage (voltam = 1) or current (2) */
ac_monV = 0.;
for (ii=0; ii < (AVE_NO / 2 + 1); ++ii)
    ac_monV += keith_rd(udvmac,Vunit,Vokay); /* in ac rms */

ac_monV /= ((REAL)(AVE_NO / 2 + 1));

if (voltam == 1)          H1 = ac_monV / ac_monR * ac_calib;
else
    {
    if ( (ac_monV >= 1.8e-4 && ac_monV <= 2.1e-4) ||
          (ac_monV >= 1.8e-3 && ac_monV <= 2.1e-3) ||
          (ac_monV >= 1.8e-2 && ac_monV <= 2.1e-2) ||
          (ac_monV >= 1.8e-1 && ac_monV <= 2.1e-1) ||
          (ac_monV >= 1.8    && ac_monV <= 2.1) )
         {
         printf("\007\007\007\nAmmeter is getting underranged,");
         printf(" reset and hit <CR>.");
         getchar();          continue;
         }
    H1 = ac_monV * ac_calib;
    }

if (dc_yesno[0] == 'y')
    {
    dc_monV = 0.;
    for (ii=0; ii < (AVE_NO / 2 + 1); ++ii)
          dc_monV += keith_rd(udvmdc,Vunit,Vokay);  /* dc Volts */

    dc_monV /= ((REAL)(AVE_NO / 2 + 1));
    Hdc = dc_monV / dc_monR * dc_calib;
    }

syn_amp = synampl(udsyn1,synunit);

if (voltam == 1)
    printf("\007\nsamp_no    ac_monV      syn_amp       Hdc(G)");
else
    printf("\007\nsamp_no    ac_monA      syn_amp       Hdc(G)");

printf("     P(%2df)(dBm)     H1(G)\n\n",harm_no);

printf("%4d  %11.5e  %9.4f%5s  %9.3e  %11.5e  %9.3e\n"
    ,samp_no,ac_monV,syn_amp,synunit,Hdc,Pnf,H1);

fprintf(outfile,
    "%4d  %9.4e  %9.4f%5s  %9.3e  %11.5e  %9.3e\n"
```

```
        ,samp_no,ac_monV,syn_amp,synunit,Hdc,Pnf,H1);

    if ((samp_no % 5) == 0)
        {
        fclose(outfile);
        outfile = fopen(filename,"a");
        }

    if (H1 > max_H1)        break;

    logamp += logstep;
    ++samp_no;
    }
ibloc(udsa1);
fprintf(outfile,"9999");
fclose(outfile);
synsetam(udsyn1,0.001);        ibloc(udsyn1);
}
```

```
/*
**      XpXppH1N.c:  This program is a newer version of
**      XpXppH1.c, which is designed to take data
**      for experiments measuring X'(1f) and X"(1f) (ac
**      susceptibility of high-Tc superconductors) vs. ac
**      magnetic field H1.
**
**      While XpXppH1.c stops after every data points to
**      allow the experimenter to manually step the ac magnetic
**      field and readjust the phase between the master and slave
**      synthesizers (syn1 & 2), this newer version automatically
**      steps H1 and take a new data point.  This new version
**      assumes that the NORMAL sine output of "syn2", a HP3325A
**      synthesizer, is driving     the ref AC input of PAR 5209 lock-in.
**
**      This program is configured by Harry Lam in August, 1990;
**      it has to be linked with par5209.c, hp3325.c, keithley.c,
**      and \gpib-pc\tc\tcibs.obj.
**
**      This program is written in Turbo C. [cf. sleep() of Turbo C.]
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include "decl.h"
#define REAL  float

int board_ud;

main()
   {
   int udsyn1,udsyn2,udlia,udvm1;
   int yesno,outquad,in_sen,out_sen;
   int samp_no,ii,jj,vv,AVE_NO,coil,mask,SAMP14;
   int read_set(),shftquad(),lisetsen();   /* Routines for lock-in */
   REAL synparam(),synphase(),synampl();   /* Routines for synthesizers */
   REAL synfreq();
   REAL rd_phase(),sig_out(),rd_timec();   /* Routines for lock-in */
   REAL keith_rd();                        /* Routines for Keithley DMM */
   REAL in_sig,out_sig,inphase,outphase,filt_f,ref_f,timec;
   REAL ac_monR,ac_calib,ac_monV,H1;
   REAL syn_ph,offset,syn_amp,syn_f,ref_min,ref_max,drv_max;
   REAL max_H1,amplit,amplit2,logamp,logstep;
   char d_res[8],f_mode[8],linefilt[8],rolloff[13],fltfunc[10];
   char osunit[5],func[10],hivolt[11],synunit[6],synfunit[4];
   char Vokay,Vunit[4],dummy[5],lastpt[10],temp[10],Hdc[10];
   char filename[20],sample[20],date[20],receiver[30];
   char overwrt[5],append,comment[80],c_yesno[5],term_yn[5];
   char NAD_yn[5];
   FILE *outfile;

   printf("\007\n\nEnter output filename:  ");
   scanf("%s",filename);
```

```
printf("\007\n\New file or OVERWRITE ? \(\"yes\" or \"no\"\)  ");
scanf("%s",overwrt);
if (overwrt[0] = = 'y' && overwrt[1] = = 'e' && overwrt[2] = = 's')
     {
     outfile = fopen(filename,"w");              append = 'n';
     }
else
     {
     outfile = fopen(filename,"a");              append = 'y';
     printf("\007\n** Appending to previously existing file. **\n");
     }

if (outfile = = NULL)
     {
     printf("File cannot be open.\n");
     exit(0);
     }

if (append = = 'n')
     {
     printf("\007\nEnter sample number (20 characters max):  ");
     scanf("%s",sample);

     printf("\007\nEnter today's date (mon-day-year):  ");
     scanf("%s",date);

     printf("\007\nEnter sample temperature (Kelvin):  ");
     scanf("%s",temp);

     printf("\007\nEnter superposing dc magnetic field (Gauss):  ");
     scanf("%s",Hdc);

     printf("\007\n(1) One-coil receiver? (2) Two-coil receiver?   ");
     scanf("%d",&coil);
     if (coil = = 1)
        sprintf(receiver,"One-coil receiver");
     else
        sprintf(receiver,"Two-coil receiver");
     }

 printf("\007\nMake sure that synth.1 & 2, lock-in and DMM are on.");
 printf("  Hit <CR> when ready.\n");
 getchar();              getchar();

 board_ud = ibfind("gpib0");
 udsyn1 = ibfind("syn1");       udsyn2 = ibfind("syn2");
 udlia = ibfind("lia");     udvm1 = keith_in("vm2");
 ibsic(board_ud);

 mask = ERR | TIMO | SRQI | RQS | CMPL;
 ibtrap(mask,2);        /* 1: off; 2: record but no trap; 3: both record
                          and trap */

 printf("\007\nWill NAD amplifier be used? ('y' or 'n') ");
```

```
scanf("%s",NAD_yn);
if (NAD_yn[0] == 'y')
      {
      printf("\007\007\nNormal range outputs are assumed ");
      printf("for BOTH synthesizers!\n");
      printf("\007\nAre BOTH syn1 ");
      printf("& syn2 outputs terminated by 50 Ohms? ('y' or 'n') ");
      scanf("%s",term_yn);
      if (term_yn[0] == 'y')
         {
         ref_min = 1.0;        ref_max = 14.;      drv_max = 4.;
         }
      else
         {
         ref_min = 0.6;        ref_max = 7.;       drv_max = 2.;
         }
      }
 else
      {
      printf("\007\nHigh voltage outputs are assumed for ");
      printf("BOTH synthesizers!\n");
      printf("\007\nEnter max. syn1's Vpp: ");
      scanf("%f",&drv_max);
      ref_min = 1.0;            ref_max = 14.;
      }

 printf("\007");
 printf("\nComputer's ready to read gen'l settings of synthesizer 1;\n");
 printf("hit <CR> if synthesizer 1 is set.\n");
 getchar();              getchar();

 offset = synparam(udsyn1,osunit,func,hivolt);
 syn_f = synfreq(udsyn1,synfunit);

 printf("\007Computer's ready to read gen'l settings of PAR5209; ");
 printf("hit <CR> if lock-in is set.\n");
 getchar();

 read_set(udlia,d_res,f_mode,&filt_f,&ref_f,linefilt,rolloff,fitfunc);
 timec = rd_timec(udlia);

 printf("\007Set the ref. phase to the IN-PHASE value, then hit <CR>.\n");
 getchar();

 inphase = rd_phase(udlia);
 in_sig = sig_out(udlia,&in_sen);

 outquad = 1;                  /* No. of quad. out-phase from in-phase */

 shftquad(udlia,outquad);      /* Default:  Shift 1 quadrant */

 printf("\007See if you like this OUT-PHASE (1),");
 printf(" or add another 180 deg (0):  ");
 scanf("%d",&yesno);
```

```
if (yesno = = 0)
      {
      shftquad(udlia,2);   /* shift 2 more quads */
      outquad = -1;
      }
outphase = rd_phase(udlia);
printf("\nin-phase = %6.2f, out-of-phase = %6.2f\n",inphase,outphase);

printf("\007\nEnter the value of the ac monitoring resistance in Ohms: ");
scanf("%f",&ac_monR);

printf("\007\nEnter calibration ot ac-coil in H1(G) / Irms(A): ");
scanf("%f",&ac_calib);

ibloc(udsyn1);
printf("\nSynthesizer 1's settings:\n");
printf("offset = %6.2f %s, %s, %s, %11.4f %s\n"
            ,offset,osunit,func,hivolt,syn_f,synfunit);
printf("\nLock-in settings:\n");
printf("%s, %s, line filt: %s, %s, %s\n"
      ,d_res,f_mode,linefilt,rolloff,fltfunc);
printf("filt. freq. = %8.3e, ref. freq. = %8.3e, ",filt_f,ref_f);
printf("timec = %7.2e s\n\n",timec);

lisetsen(udlia,in_sen);
shftquad(udlia,-outquad);

printf("\007How many points to average? ");
scanf("%d",&AVE_NO);

if (append = = 'n')
      {
      printf("\007\n");
      printf("Want to add some comment ( < 80 char, 'y' or 'n')? ");
      scanf("%s",c_yesno);
      if (c_yesno[0] = = 'y')
         {
         getchar();
         printf("Comment: "); gets(comment);
         }

      fprintf(outfile,"File %s\n",filename);
      fprintf(outfile,"Date: %s,  Sample: %s,  Temp: %s K,  Hdc: %s G\n"
         ,date,sample,temp,Hdc);
      fprintf(outfile,"%s\n",receiver);
      fprintf(outfile,"\nSynthesizer 1 (slave) settings:\n");
      fprintf(outfile,"offset = %6.2f %s, %s, %s, %11.4f %s\n"
            ,offset,osunit,func,hivolt,syn_f,synfunit);
      fprintf(outfile,"\nLock-in settings:\n");
      fprintf(outfile,"%s, %s, line filt: %s, %s, %s\n"
            ,d_res,f_mode,linefilt,rolloff,fltfunc);
      fprintf(outfile,"filt. freq. = %8.3e, ref. freq. = %8.3e, "
            ,filt_f,ref_f);
      fprintf(outfile,"timec = %7.2e s\n\n",timec);
```

```
        fprintf(outfile,"In-phase = %6.2f, out-of-phase = %6.2f\n
                ,inphase,outphase);

        fprintf(outfile,"Ac monitoring resistance in Ohms: %7.3f\n\n"
                ,ac_monR);

        fprintf(outfile,
                "Calibration of ac-coil in H1(G) / Irms(A): %8.3f\n\n"
                ,ac_calib);

        fprintf(outfile,"# of ave. / data point = %d\n\n",AVE_NO);

        if (c_yesno[0] == 'y')    fprintf(outfile,"%s\n\n",comment);

        fprintf(outfile,"*****\n");
        fprintf(outfile," No    ac_monV      syn_amp      syn_ph");
        fprintf(outfile," in_sig(mV)   out_sig(mV)     H1(G)\n\n");
        }

printf("\007\007\007\007\n");
printf("About to start taking data, ensure ");
printf("ref. phase of lock-in is IN-PHASE (%6.1f)!\n\n",inphase);

ibwrt(udsyn1,"AM",2);    ibloc(udsyn1);
printf("\007Input the starting syn1 ampltude (volts): ");
scanf("%f",&amplit);
logamp = log10(amplit);
printf("\007Input max. H1 desired: ");
scanf("%f",&max_H1);
printf("\007Input no. of samples per decade: ");
scanf("%d",&SAMP14);

printf("\007\nHit <CR> when ready. (IN-PHASE %7.2f first)",inphase);
getchar();    getchar();

logstep = 1. / ((REAL)SAMP14);   /* samples per decade */
samp_no = 1;
for ( ; ; )
     {
     amplit = pow(10.,logamp);
     if (amplit > drv_max)
        {
        ibloc(udsyn1);
        printf("\007\007\007\nMANUALLY reset NAD amplifier ");
        printf("input level and synth. 1 amplitude.\n");
        printf("\nEnter new syn1 amplitude ");
        printf("(in volts, or 99999 to quit): ");
        scanf("%f",&amplit);
        if (amplit > drv_max && amplit < = 40.)
             {
             printf("\007\007\007\nEntry");
             printf(" larger than %6.2f volts, reenter to reconfirm: "
                     ,drv_max);
             scanf("%f",&amplit);
```

```
        }
    logamp = log10(amplit);
    if (amplit > 40.)    break;
    }

/* Choose a corresponding ac output of the master synthesizer,
   which is driving the "AC IN" of the lock-in reference, so
   that the phase of the reference will change correspondingly
   with that of the slave synthesizer, which is driving the
   ac magnetic field. */

amplit2 = amplit;
if (amplit2 < ref_min)    /* too small for lock-in ref */
    {
    for (jj=0; jj<4; ++jj)
        {
        amplit2 *= 10.;
        if (amplit2 >= ref_min)          break;
        }
    }
else if (amplit2 > ref_max)  /* too large for lock-in ref */
    {
    for (jj=0; jj<2; ++jj)
        {
        amplit2 /= 10.;
        if (amplit2 <= ref_max)                  break;
        }
    }

synsetam(udsyn1,amplit);    synsetam(udsyn2,amplit2);

lkinwait(udlia);

in_sig = 0.;
for (ii=0; ii<AVE_NO; ++ii)    in_sig += sig_out(udlia,&in_sen);
in_sig /= ((REAL) AVE_NO);

if (samp_no == 1)        out_sen = in_sen;

/* prevent overload when switch phase */
lisetsen(udlia,15);              sleep(1);    /* Turbo C */
shftquad(udlia,outquad);    lisetsen(udlia,out_sen);

lkinwait(udlia);

out_sig = 0.;                                /*in millivolts */
for (ii=0; ii<AVE_NO; ++ii)    out_sig += sig_out(udlia,&out_sen);
out_sig /= ((REAL) AVE_NO);

/* Read synthesizer 2's phase and ac monitor voltage */
ac_monV = 0.;
for (ii=0; ii < (AVE_NO / 2 + 1); ++ii)
    ac_monV += keith_rd(udvm1,Vunit,Vokay); /* in ac Volts rms */
```

```
    ac_monV /= ((REAL)(AVE_NO / 2 + 1));

    syn_ph = synphase(udsyn2);                  /* in degrees */
    syn_amp = synampl(udsyn1,synunit);

    H1 = ac_monV / ac_monR * ac_calib;

    /* prevent overload when switch phase */
    lisetsen(udlia,15);          sleep(1);  /* Turbo C */
    shftquad(udlia,-outquad);          lisetsen(udlia,in_sen);

    printf("\007\nsamp_no   ac_monV    syn_amp     syn_ph");
    printf("   in_sig     out_sig     H1(G)\n\n");

    printf("%4d  %11.5e  %6.4f%5s  %6.1f  %9.3e  %9.3e  %9.3e\n"
       ,samp_no,ac_monV,syn_amp,synunit,syn_ph,in_sig,out_sig,H1);

    fprintf(outfile,
       "%4d  %9.4e  %6.4f%5s  %6.1f  %10.4e  %10.4e  %9.3e\n"
       ,samp_no,ac_monV,syn_amp,synunit,syn_ph,in_sig,out_sig,H1);

    if ((samp_no % 5) == 0)
       {
       fclose(outfile);
       outfile = fopen(filename,"a");
       }

    if (H1 > max_H1)        break;

    logamp += logstep;
    ++samp_no;
    }
ibloc(udsyn1);          ibloc(udsyn2);
fprintf(outfile,"9999");
fclose(outfile);
}
```

```
/*
**      XpXppH1T.c:  This program is another modified
**      version of XpXppH1.c, which is designed to take data
**      for experiments measuring X'(1f) and X"(1f) (of
**      high-Tc superconductors) vs. ac magnetic field H1.
**
**      This program is modified from XpXppH1n.c so that
**      Keithley 197 DMM (device name: "vm1") will be used to
**      monitor the thermocouple voltage, hence the sample
**      temperature.
**
**      This program is configured by Harry Lam in March, 1991;
**      it has to be linked to par5209.c, hp3325.c, keithley.c,
**      and \gpib-pc\tc\tcibs.obj.
*/

#include <stdio.h>
#include <math.h>
#include <dos.h>
#include "decl.h"
#define REAL  float

int board_ud;

main()
   {
   int udsyn1,udsyn2,udlia,udvm1,udvm2;
   int yesno,outquad,in_sen,out_sen;
   int samp_no,ii,jj,vv,AVE_NO,coil,mask,SAMP14;
   int read_set(),shftquad(),lisetsen();  /* Routines for lock-in */
   REAL synparam(),synphase(),synampl();  /* Routines for synthesizers */
   REAL synfreq();
   REAL rd_phase(),sig_out(),rd_timec();  /* Routines for lock-in */
   REAL keith_rd();                       /* Routines for Keithley DMM */
   REAL in_sig,out_sig,inphase,outphase,filt_f,ref_f,timec;
   REAL ac_monR,ac_calib,ac_monV,H1,TCV;
   REAL offset,syn_amp,syn_f,ref_min,ref_max,drv_max;
   REAL max_H1,logamp,logstep;
   double amplit,amplit2;
   char d_res[8],f_mode[8],linefilt[8],rolloff[13],fltfunc[10];
   char osunit[5],func[10],hivolt[11],synunit[6],synfunit[4];
   char Vokay,Vunit[4],dummy[5],lastpt[10],temp[10],Hdc[10];
   char filename[20],sample[20],date[20],receiver[30];
   char overwrt[5],append,comment[80],c_yesno[5],term_yn[5];
   char NAD_yn[5];
   FILE *outfile;

   printf("\007\n\nEnter output filename:  ");
   scanf("%s",filename);

   printf("\007\n\New file or OVERWRITE ? \(\"yes\" or \"no\"\)  ");
   scanf("%s",overwrt);
   if (overwrt[0] = = 'y' && overwrt[1] = = 'e' && overwrt[2] = = 's')
        {
```

```
      outfile = fopen(filename,"w");                append = 'n';
      }
else
      {
      outfile = fopen(filename,"a");                append = 'y';
      printf("\007\n** Appending to previously existing file. **\n");
      }

if (outfile == NULL)
      {
      printf("File cannot be open.\n");
      exit(0);
      }

if (append == 'n')
      {
      printf("\007\nEnter sample number (20 characters max): ");
      scanf("%s",sample);

      printf("\007\nEnter today's date (mon-day-year): ");
      scanf("%s",date);

      printf("\007\nEnter sample temperature (Kelvin): ");
      scanf("%s",temp);

      printf("\007\nEnter superposing dc magnetic field (Gauss): ");
      scanf("%s",Hdc);

      printf("\007\n(1) One-coil receiver? (2) Two-coil receiver?  ");
      scanf("%d",&coil);
      if (coil == 1)
         sprintf(receiver,"One-coil receiver");
      else
         sprintf(receiver,"Two-coil receiver");
      }

 printf("\007\nMake sure that synth.1 & 2, lock-in, vm1, vm2 are on.");
 printf("  Hit <CR> when ready.\n");
 getchar();              getchar();

 board_ud = ibfind("gpib0");
 udsyn1 = ibfind("syn1");       udsyn2 = ibfind("syn2");
 udlia = ibfind("lia");     udvm2 = keith_in("vm2");
 udvm1 = keith_in("vm1");
 ibsic(board_ud);

 mask = ERR | TIMO | SRQI | RQS | CMPL;
 ibtrap(mask,1);        /* 1: off; 2: record but no trap; 3: both record
                          and trap */

 printf("\007\nWill NAD amplifier be used? ('y' or 'n') ");
 scanf("%s",NAD_yn);
 if (NAD_yn[0] == 'y')
       {
```

```
        printf("\007\007\nNormal range outputs are assumed ");
        printf("for BOTH synthesizers!\n");
        printf("\007\nAre BOTH syn1 ");
        printf("& syn2 outputs terminated by 50 Ohms? ('y' or 'n') ");
        scanf("%s",term_yn);
        if (term_yn[0] = = 'y')
           {
           ref_min = 1.0;        ref_max = 14.;      drv_max = 4.;
           }
        else
           {
           ref_min = 0.6;        ref_max = 7.;       drv_max = 2.;
           }
        }
else
        {
        printf("\007\nHigh voltage outputs are assumed for ");
        printf("BOTH synthesizers!\n");
        printf("\007\nEnter max. syn1's Vpp: ");
        scanf("%f",&drv_max);
        ref_min = 1.0;            ref_max = 14.;
        }

printf("\007");
printf("\nComputer's ready to read gen'l settings of synthesizer 1;\n");
printf("hit <CR> if synthesizer 1 is set.\n");
getchar();              getchar();

offset = synparam(udsyn1,osunit,func,hivolt);
syn_f = synfreq(udsyn1,synfunit);

printf("\007Computer's ready to read gen'l settings of PAR5209; ");
printf("hit <CR> if lock-in is set.\n");
getchar();

read_set(udlia,d_res,f_mode,&filt_f,&ref_f,linefilt,rolloff,fltfunc);
timec = rd_timec(udlia);

printf("\007Set the ref. phase to the IN-PHASE value, then hit <CR>.\n");
getchar();

inphase = rd_phase(udlia);
in_sig = sig_out(udlia,&in_sen);

outquad = 1;                    /* No. of quad. out-phase from in-phase */

shftquad(udlia,outquad);        /* Default:  Shift 1 quadrant */

printf("\007See if you like this OUT-PHASE (1),");
printf(" or add another 180 deg (0):  ");
scanf("%d",&yesno);

if (yesno = = 0)
     {
```

```
        shftquad(udlia,2);   /* shift 2 more quads */
        outquad = -1;
        }
outphase = rd_phase(udlia);
printf("\nin-phase = %6.2f, out-of-phase = %6.2f\n",inphase,outphase);

printf("\007\nEnter the value of the ac monitoring resistance in Ohms:  ");
scanf("%f",&ac_monR);

printf("\007\nEnter calibration of ac-coil in H1(G) / Irms(A):  ");
scanf("%f",&ac_calib);

ibloc(udsyn1);
printf("\nSynthesizer 1's settings:\n");
printf("offset = %6.2f %s, %s, %s, %11.4f %s\n"
                ,offset,osunit,func,hivolt,syn_f,synfunit);
printf("\nLock-in settings:\n");
printf("%s, %s, line filt: %s, %s, %s\n"
        ,d_res,f_mode,linefilt,rolloff,fltfunc);
printf("filt. freq. = %8.3e, ref. freq. = %8.3e, ",filt_f,ref_f);
printf("timec = %7.2e s\n\n",timec);

lisetsen(udlia,in_sen);
shftquad(udlia,-outquad);

printf("\007How many points to average?  ");
scanf("%d",&AVE_NO);

if (append = = 'n')
        {
        printf("\007\n");
        printf("Want to add some comment ( < 80 char, 'y' or 'n')?  ");
        scanf("%s",c_yesno);
        if (c_yesno[0] = = 'y')
            {
            getchar();
            printf("Comment: ");  gets(comment);
            }

        fprintf(outfile,"File %s\n",filename);
        fprintf(outfile,"Date: %s,  Sample: %s,  Temp: %s K,  Hdc: %s G\n"
            ,date,sample,temp,Hdc);
        fprintf(outfile,"%s\n",receiver);
        fprintf(outfile,"\nSynthesizer 1 (slave) settings:\n");
        fprintf(outfile,"offset = %6.2f %s, %s, %s, %11.4f %s\n"
                ,offset,osunit,func,hivolt,syn_f,synfunit);
        fprintf(outfile,"\nLock-in settings:\n");
        fprintf(outfile,"%s, %s, line filt: %s, %s, %s\n"
                ,d_res,f_mode,linefilt,rolloff,fltfunc);
        fprintf(outfile,"filt. freq. = %8.3e, ref. freq. = %8.3e, "
                ,filt_f,ref_f);
        fprintf(outfile,"timec = %7.2e s\n\n",timec);

        fprintf(outfile,"In-phase = %6.2f, out-of-phase = %6.2f\n"
```

```
            ,inphase,outphase);

      fprintf(outfile,"Ac monitoring resistance in Ohms:  %7.3f\n\n"
            ,ac_monR);

      fprintf(outfile,
            "Calibration of ac-coil in H1(G) / Irms(A):  %8.3f\n\n"
            ,ac_calib);

      fprintf(outfile,"# of ave. / data point = %d\n\n",AVE_NO);

      if (c_yesno[0] = = 'y')    fprintf(outfile,"%s\n\n",comment);

      fprintf(outfile,"*****\n");
      fprintf(outfile,"  No    ac_monV      syn_amp       TC(V)");
      fprintf(outfile,"   in_sig(mV)    out_sig(mV)     H1(G)\n\n");
      }

printf("\007\007\007\007\n");
printf("About to start taking data, ensure ");
printf("ref. phase of lock-in is IN-PHASE (%6.1f)!\n\n",inphase);

ibwrt(udsyn1,"AM",2);    ibloc(udsyn1);
printf("\007Input the starting syn1 ampltude (volts):  ");
scanf("%lf",&amplit);
logamp = log10(amplit);
printf("\007Input max. H1 desired:  ");
scanf("%f",&max_H1);
printf("\007Input no. of samples per decade:  ");
scanf("%d",&SAMP14);

printf("\007\nHit <CR> when ready. (IN-PHASE %7.2f first)",inphase);
getchar();    getchar();

logstep = 1. / ((REAL)SAMP14);   /* samples per decade */
samp_no = 1;
for ( ; ; )
     {
     amplit = pow(10.,logamp);
     if (amplit > drv_max)
        {
        ibloc(udsyn1);
        printf("\007\007\007\nMANUALLY reset NAD amplifier ");
        printf("input level and synth. 1 amplitude.\n");
        printf("\nEnter new syn1 amplitude ");
        printf("(in volts, or 99999 to quit):  ");
        scanf("%lf",&amplit);
        if (amplit > drv_max && amplit < = 40.)
             {
             printf("\007\007\007\nEntry");
             printf(" larger than %6.2f volts, reenter to reconfirm: "
                  ,drv_max);
             scanf("%lf",&amplit);
             }
```

```
   logamp = log10(amplit);
   if (amplit > 40.)    break;
   }

amplit2 = amplit;
if (amplit2 < ref_min)    /* too small for lock-in ref */
   {
   for (jj=0; jj<4; ++jj)
        {
        amplit2 *= 10.;
        if (amplit2 >= ref_min)          break;
        }
   }
else if (amplit2 > ref_max)  /* too large for lock-in ref */
   {
   for (jj=0; jj<2; ++jj)
        {
        amplit2 /= 10.;
        if (amplit2 <= ref_max)                  break;
        }
   }

synsetam(udsyn1,amplit);    synsetam(udsyn2,amplit2);

lkinwait(udlia);

in_sig = 0.;
for (ii=0; ii<AVE_NO; ++ii)    in_sig += sig_out(udlia,&in_sen);
in_sig /= ((REAL) AVE_NO);

if (samp_no == 1)       out_sen = in_sen;

/* prevent overload when switch phase */
lisetsen(udlia,15);          sleep(1);    /* Turbo C */
shftquad(udlia,outquad);    lisetsen(udlia,out_sen);

lkinwait(udlia);

out_sig = 0.;                                   /*in millivolts */
for (ii=0; ii<AVE_NO; ++ii)    out_sig += sig_out(udiia,&out_sen);
out_sig /= ((REAL) AVE_NO);

/* Read synthesizer 2's phase and ac monitor voltage */
ac_monV = 0.;
for (ii=0; ii < (AVE_NO / 2 + 1); ++ii)
   ac_monV += keith_rd(udvm2,Vunit,Vokay); /* in ac Volts rms */

ac_monV /= ((REAL)(AVE_NO / 2 + 1));

TCV = keith_rd(udvm1,Vunit,Vokay);
syn_amp = synampl(udsyn1,synunit);

H1 = ac_monV / ac_monR * ac_calib;
```

```
    /* prevent overload when switch phase */
    lisetsen(udlia,15);          sleep(1);  /* Turbo C */
    shftquad(udlia,-outquad);          lisetsen(udlia,in_sen);

    printf("\007\nsamp_no   ac_monV    syn_amp      TC(V)");
    printf("   in_sig    out_sig    H1(G)\n\n");

    printf("%3d %11.5e %10.4e%5s %11.5e %9.3e %9.3e %9.3e\n"
       ,samp_no,ac_monV,syn_amp,synunit,TCV,in_sig,out_sig,H1);

    fprintf(outfile,
       "%3d %11.5e %10.4e%5s %11.5e %10.4e %10.4e %9.3e\n"
       ,samp_no,ac_monV,syn_amp,synunit,TCV,in_sig,out_sig,H1);

    if ((samp_no % 3) == 0)
       {
       fclose(outfile);
       outfile = fopen(filename,"a");
       }

    if (H1 > max_H1)       break;

    logamp += logstep;
    ++samp_no;
    }
ibloc(udsyn1);          ibloc(udsyn2);
fprintf(outfile,"9999");
fclose(outfile);
}
```

```
/*
**   pltspc.c:  This program reads the data written by sadump.c
**   and output them to HP7225B.
**
**   This program has to be linked with hpgraph.c, and
**   \gpib-pc\tc\tcibs.obj.
*/

#include <stdio.h>
#define REAL float
main()

{
   REAL spec[1001],tckx,tcky,freq,ispec;
   REAL y_pos,reflev,botlev;
   FILE *file;
   char ref[15],mkrfrq[26],dbdiv[10],range[16],mkramp[15];
   char ctrfrq[23],span[21],rbw[12],vbw[12],st[13];
   char string[100],aspec[6],afreq[6];
   int ud,ii,jj,kk;
   extern uudd;

   printf("Enter input filename:  ");
   scanf("%s",string);

   file = fopen(string,"r");

   for (ii=0; ii<14; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",ref+0);
        else     */   fscanf(file,"%c",ref+ii);
        }
   fscanf(file,"%*c");
   ref[14] = '\0';

   for (ii=0; ii<25; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",mkrfrq+0);
        else*/      fscanf(file,"%c",mkrfrq+ii);
        }
   fscanf(file,"%*c");
   mkrfrq[25] = '\0';

   for (ii=0; ii<9; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",dbdiv+0);
        else*/      fscanf(file,"%c",dbdiv+ii);
        }
   fscanf(file,"%*c");
   dbdiv[9] = '\0';

   for (ii=0; ii<15; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",range+0);
```

```
        else */     fscanf(file,"%c",range+ii);
        }
    fscanf(file,"%*c");
    range[15] = '\0';

    for (ii=0; ii<14; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",mkramp+0);
        else    */  fscanf(file,"%c",mkramp+ii);
        }
    fscanf(file,"%*c");
    mkramp[14] = '\0';

    for (ii=0; ii<22; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",ctrfrq+0);
        else */     fscanf(file,"%c",ctrfrq+ii);
        }
    fscanf(file,"%*c");
    ctrfrq[22] = '\0';

    for (ii=0; ii<20; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",span+0);
        else    */  fscanf(file,"%c",span+ii);
        }
    fscanf(file,"%*c");
    span[20] = '\0';

    for (ii=0; ii<11; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",rbw+0);
        else*/      fscanf(file,"%c",rbw+ii);
        }
    fscanf(file,"%*c");
    rbw[11] = '\0';

    for (ii=0; ii<11; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",vbw+0);
        else*/      fscanf(file,"%c",vbw+ii);
        }
    fscanf(file,"%*c");
    vbw[11] = '\0';

    for (ii=0; ii<12; ++ii)
        {
/*      if (ii==0) fscanf(file,"%1s",st+0);
        else*/      fscanf(file,"%c",st+ii);
        }
    fscanf(file,"%*c");
    st[12] = '\0';

    sscanf(ref,"%*4c%f",&reflev);
```

```
   botlev = reflev - 100.;

/*    printf("REF = %f\n",reflev);
   printf("%14s, %25s, %9s\n",ref,mkrfrq,dbdiv);
   printf("%15s, %14s, %22s\n",range,mkramp,ctrfrq);
   printf("%20s, %11s, %11s, %12s\n",span,rbw,vbw,st);
 */
   for (ii=0; ii<1001; ++ii)
        {
        fscanf(file,"%e,",spec+ii);
/*      if (ii < 5 || ii > 996)
           {
           if (ii == 997)          printf("\n");
           printf("%9.3le ",spec[ii]);
           }
*/
        }

   fclose(file);
/*   getchar();
   getchar(); */

   ud = initgraph("PL1");
   clear();

   scale(0.,10000.,botlev,reflev);
   border();

   axes(0.,botlev,1000.,10.,1,1);
   axes(10000.,reflev,1000.,10.,1,1);

   freq = 0.;
   for (ii=0; ii<1001; ++ii)
        {
        ispec = spec[ii];
        move(freq,ispec);
        if (ii == 0)               pendown();
        if (ii == 1000)  penup();
        freq += 10.;
        }

   y_pos = reflev + 5.;
   move(4000.,y_pos);    charsize(.2,.4);
   label(string);
   y_pos = botlev + 40.;
   labeldir(90.); move(-500.,y_pos);
   label("Power (dBm)");          labeldir(0.);
   y_pos = botlev - 5.;
   move(0.,y_pos);      charsize(.15,.3);
   label(ref);              label(",");
   label(mkrfrq);           label(",");
   label(dbdiv); label(",");
   label(range); label(",");          label("\015\012");
   label(mkramp);         label(",");
```

```
    label(ctrfrq); label(",");
    label(span);   label(",");
    label(rbw);              label(",");          label("\015\012");
    label(vbw);              label(",");
    label(st);
}
```

```
/*
**    plotpnf.c:    Turbo C video graphics program to
**    read a data file generated by experiments controlled
**    by Pnf_H1.c and plot the results on the video.
*/

#include <stdio.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <process.h>
#define  NO_LINES  27
#define  REAL          float
#define  ENDFILE   9999
#define  PI  3.141592654

main()
   {
   char line[NO_LINES][80],filename[20],synampU[5],label[30];
   char normyn[5];
   REAL ac_monV,syn_amp,Pnf.H1,Hdc;
   REAL logH1,yplot;
   REAL xmin,xmax,xtic,ymin,ymax,ytic,labelx,labely;
   int ii,jj,textline=0,samp_no;
   int in_out;
   FILE *fileptr;

   printf("Enter input filename:  ");
   scanf("%s",filename);
   fileptr = fopen(filename,"r");

   for (ii=0; ii<NO_LINES-1; ++ii)
        {
        fgets(line[ii],80,fileptr);

        for (jj=0; jj<4; ++jj)
           {
           if (line[ii][jj] != '*')
                {
                if (ii < (NO_LINES - 2))
                   break;
                else
                   {
                   printf("Not enough memory assigned for text.\n");
                   exit(0);
                   }
                }
           if (jj == 3)  textline = ii + 1;
           }

        if (textline == (ii+1))              break;
        }

   fgets(line[textline],80,fileptr);      /* Column headings. */
```

```
    fgets(line[textline-1],80,fileptr);          /* Blank line replaces *'s. */

    printf("Normalized? ('y' or 'n')  ");
    scanf("%s",normyn);
    printf("Enter xmin, xmax, xtic, ymin, ymax and ytic:\n");
    scanf("%f %f %f %f %f %f",&xmin,&xmax,&xtic,&ymin,&ymax,&ytic);

    v_initgraph();
/*    v_aspect();  */
    v_scale(xmin,xmax,ymin,ymax);
    v_border();
    v_axes(xmin,ymin,xtic,ytic,1,1);
    v_axes(xmax,ymax,xtic,ytic,1,1);
    v_logxaxes(xmin,xmax,ymin,ymax,1);
    labelx = xmin + (xmax - xmin) * .1;
    labely = ymax + (ymax - ymin) * .04;
    v_move(labelx,labely);
    v_label(line[0]);
    labelx = xmin - (xmax - xmin) * 0.04;
    labely = ymin - (ymax - ymin) * 0.06;
    v_move(labelx,labely);
    sprintf(label,"%3.0f",xmin);
    v_label(label);
    labelx = xmax - (xmax - xmin) * 0.04;
    v_move(labelx,labely);
    sprintf(label,"%3.0f",xmax);
    v_label(label);
    labelx = xmin + (xmax - xmin) * .48;
    labely = ymin - (ymax - ymin) * 0.08;
    v_move(labelx,labely);
    sprintf(label,"Log (H1)");
    v_label(label);
    labelx = xmin - (xmax - xmin) * .04;
    labely = ymin + (ymax - ymin) * .25;
    v_move(labelx,labely);          v_textdir(1);
    if (normyn[0] == 'y')
        sprintf(label,"P(nf) / (H1 x H1) (dB)");
    else
        sprintf(label,"    P(nf)  (dBm)");

    v_label(label);
    labelx = xmin - (xmax - xmin) * .15;
    labely = ymin;
    v_move(labelx,labely);          v_textdir(0);
    sprintf(label,"%9.3e",ymin);
    v_label(label);
    labelx = xmin - (xmax - xmin) * .15;
    labely = ymax;
    v_move(labelx,labely);
    sprintf(label,"%9.2e",ymax);
    v_label(label);

    for (ii = 1; ii < 1000; ++ii)
        {
```

```
        fscanf(fileptr,"%d ",&samp_no);
        if (samp_no = = ENDFILE)                    break;
        fscanf(fileptr,"%e %f %s %e %e %e"
          ,&ac_monV,&syn_amp,synampU,&Hdc,&Pnf,&H1);

        logH1 = log10(H1);          yplot = Pnf;

        if (normyn[0] = = 'y')      yplot -= (20. * logH1);

        if (ii = = 1)               v_move(logH1,yplot);
        else                        v_lineto(logH1,yplot);
        }

fclose(fileptr);
getchar():          getchar();

v_initgraph();
v_scale(xmin,xmax,ymin,ymax);
labelx = xmin - (xmax - xmin) * 0.125;
labely = ymax;
for (ii=0; ii < textline; ++ii)
      {
      v_move(labelx,labely);
      v_label(line[ii]);
      labely -= ((ymax - ymin) * 0.06);
      }
getchar();                  getchar();
clearviewport();            closegraph();
}
```

```
/*
**    hplotpnf.c:    C graphics program to read a data
**    file generated by experiments controlled by
**    Pnf_H1.c and plot the results on the HP7225B plotter.
*/

#include <stdio.h>
#include <math.h>
#include "decl.h"          /* for GPIB NI-488 handler */
#define  NO_LINES  27
#define  REAL          float
#define  ENDFILE   9999
#define  PI  3.141592654

main()
   {
   char line[NO_LINES][80],filename[20],synampU[5],labelstr[30];
   char normyn[5],correct[5];
   REAL ac_monV,syn_amp,Pnf,H1,Hdc;
   REAL logH1,yplot,radius;
   REAL xmin,xmax,xtic,ymin,ymax,ytic,labelx,labely,factor;
   int ii,jj,textline=0,samp_no;
   int in_out,linecirc,ud,every,cangle;
   FILE *fileptr;

   printf("Enter input filename:  ");
   scanf("%s",filename);
   fileptr = fopen(filename,"r");

   for (ii=0; ii<NO_LINES-1; ++ii)
        {
        fgets(line[ii],80,fileptr);

        for (jj=0; jj<4; ++jj)
           {
           if (line[ii][jj] != '*')
                {
                if (ii < (NO_LINES - 2))
                   break;
                else
                   {
                   printf("Not enough memory assigned for text.\n");
                   exit(0);
                   }
                }
           if (jj == 3)  textline = ii + 1;
           }

        if (textline == (ii+1))          break;
        }

    fgets(line[textline],80,fileptr);      /* Column headings. */
    fgets(line[textline-1],80,fileptr);          /* Blank line replaces *'s. */
```

```
printf("Normalized? ('y' or 'n')  ");
scanf("%s",normyn);
printf("Want to a correct. factor to H1 orig. calib.? ('y' or 'n')  ");
scanf("%s",correct);
if (correct[0] = = 'y')
      {
      printf("Enter correction factor:  ");
      scanf("%f",&factor);
      }
else  factor = 1.;

printf("1. Line;  2. Circle;  3. Diamond ?  ");
scanf("%d",&linecirc);
printf("Enter xmin, xmax, xtic, ymin, ymax and ytic:\n");
scanf("%f %f %f %f %f %f",&xmin,&xmax,&xtic,&ymin,&ymax,&ytic);
if (linecirc = = 2 || linecirc = = 3)
      {
      radius = (xmax - xmin) / 150.;

      if (linecirc = = 2)            cangle = 30;
      else                           cangle = 90;

      printf("Plot  1. every data pt.;  2. every OTHER pt.?  ");
      scanf("%d",&every);
      }

ud = initgraph("PL1");          clear();
scale(xmin,xmax,ymin,ymax);
border();
axes(xmin,ymin,xtic,ytic,1,1);
axes(xmax,ymax,xtic,ytic,1,1);
log_xaxes(xmin,xmax,ymin,ymax,1);
labelx = xmin + (xmax - xmin) * .1;
labely = ymax + (ymax - ymin) * .04;
move(labelx,labely);
label(line[0]);
labelx = xmin - (xmax - xmin) * 0.04;
labely = ymin - (ymax - ymin) * 0.06;
move(labelx,labely);
sprintf(labelstr,"%3.0f",xmin);
label(labelstr);
labelx = xmax - (xmax - xmin) * 0.04;
move(labelx,labely);
sprintf(labelstr,"%3.0f",xmax);
label(labelstr);
labelx = xmin + (xmax - xmin) * .48;
labely = ymin - (ymax - ymin) * 0.08;
move(labelx,labely);
sprintf(labelstr,"Log (H1)");
label(labelstr);
labelx = xmin - (xmax - xmin) * .04;
labely = ymin + (ymax - ymin) * .25;
move(labelx,labely);   labeldir(90.);
if (normyn[0] = = 'y')
```

```
        sprintf(labelstr,"P(nf) / (H1 x H1) (dB)");
    else
        sprintf(labelstr,"    P(nf)  (dBm)");

    label(labelstr);
    labelx = xmin - (xmax - xmin) * .15;
    labely = ymin;
    move(labelx,labely);   labeldir(0.);
    sprintf(labelstr,"%9.3e",ymin);
    label(labelstr);
    labelx = xmin - (xmax - xmin) * .15;
    labely = ymax;
    move(labelx,labely);
    sprintf(labelstr,"%9.2e",ymax);
    label(labelstr);

    for (ii=1; ii<1000; ++ii)
        {
        fscanf(fileptr,"%d ",&samp_no);
        if (samp_no == ENDFILE)                 break;
        fscanf(fileptr,"%e %f %s %e %e %e"
          ,&ac_monV,&syn_amp,synampU,&Hdc,&Pnf,&H1);

        H1 *= factor;
        logH1 = log10(H1);        yplot = Pnf;

        if (normyn[0] == 'y')       yplot -= (20. * logH1);

        if (linecirc == 1)
            {
            move(logH1,yplot);
            if (ii == 1)        pendown();
            }
        else
            {
            if ( every == 1 || (ii % 2) == 0 )
                circleat(logH1,yplot,radius,cangle);
            }
        }

    penup();
    fclose(fileptr);

/*  getchar();          getchar();

    v_initgraph();
    v_scale(xmin,xmax,ymin,ymax);
    labelx = xmin - (xmax - xmin) * 0.125;
    labely = ymax;
    for (ii=0; ii < textline; ++ii)
        {
        v_move(labelx,labely);
        v_label(line[ii]);
        labely -= ((ymax - ymin) * 0.06);
```

```
        }
    getchar();              getchar();
    clearviewport();        closegraph();
*/
    }
```

```
/*
**    vplotf1.c:    Turbo C video graphics program to
**    read a data file generated by experiments controlled
**    by XpXppH1.c or XpXppH1n.c, and plot the results on the video.
**
**    This program has to be linked to v_graph.c.
*/

#include <stdio.h>
#include <math.h>
#include <graphics.h>
#include <conio.h>
#include <process.h>
#define  NO_LINES  25
#define  REAL          float
#define  ENDFILE   9999
#define  PI  3.141592654

main()
   {
   char line[NO_LINES][80],filename[20],synampU[5],label[20];
   REAL ac_monV,syn_amp,syn_ph,in_sig,out_sig,H1;
   REAL logH1,radius,yplot;
   REAL xmin,xmax,xtic,ymin,ymax,ytic,labelx,labely;
   REAL fr_angle,sinfr,cosfr;
   int size = 80,ii,jj,textline=0,samp_no;
   int in_out;
   FILE *fileptr;

   printf("Enter input filename:  ");
   scanf("%s",filename);
   fileptr = fopen(filename,"r");

   for (ii=0; ii<NO_LINES-1; ++ii)
        {
        fgets(line[ii],80,fileptr);

        for (jj=0; jj<4; ++jj)
           {
           if (line[ii][jj] != '*')
                {
                if (ii < (NO_LINES - 2))
                   break;
                else
                   {
                   printf("Not enough memory assigned for text.\n");
                   exit(0);
                   }
                }
           if (jj == 3)  textline = ii + 1;
           }

        if (textline == (ii+1))              break;
        }
```

```
    fgets(line[textline],80,fileptr);       /* Column headings. */
    fgets(line[textline-1],80,fileptr);             /* Blank line replaces *'s. */

    printf("(1) V'(nf) / H1 ?   (2) V\"(nf) / H1 ?   ");
    printf("(3) P(nf) in dBm ?  ");
    scanf("%d",&in_out);

    if (in_out == 1 || in_out == 2)
         {
         printf("Enter frame rotation angle in degrees:  ");
         scanf("%f",&fr_angle);
         }
    else
         fr_angle = 0.;

    fr_angle *= (PI / 180.);
    sinfr = sin(fr_angle); cosfr = cos(fr_angle);

    printf("Enter xmin, xmax, xtic, ymin, ymax and ytic:\n");
    scanf("%f %f %f %f %f %f",&xmin,&xmax,&xtic,&ymin,&ymax,&ytic);
    radius = (xmax - xmin) * 0.01;

    v_initgraph();
/*  v_aspect();  */
    v_scale(xmin,xmax,ymin,ymax);
    v_border();
    v_axes(xmin,ymin,xtic,ytic,1,1);
    v_axes(xmax,ymax,xtic,ytic,1,1);
    v_logxaxes(xmin,xmax,ymin,ymax,1);
    labelx = xmin + (xmax - xmin) * .1;
    labely = ymax + (ymax - ymin) * .04;
    v_move(labelx,labely);
    v_label(line[0]);
    labelx = xmin - (xmax - xmin) * 0.04;
    labely = ymin - (ymax - ymin) * 0.06;
    v_move(labelx,labely);
    sprintf(label,"%3.0f",xmin);
    v_label(label);
    labelx = xmax - (xmax - xmin) * 0.04;
    v_move(labelx,labely);
    sprintf(label,"%3.0f",xmax);
    v_label(label);
    labelx = xmin + (xmax - xmin) * .48;
    labely = ymin - (ymax - ymin) * 0.08;
    v_move(labelx,labely);
    sprintf(label,"Log (H1)");
    v_label(label);
    labelx = xmin - (xmax - xmin) * .04;
    labely = ymin + (ymax - ymin) * .25;
    v_move(labelx,labely);          v_textdir(1);

    if (in_out == 1)
         sprintf(label,"V'(nf) / H1 (mV/Oe)");
    else if (in_out == 2)
```

```
    sprintf(label,"V\"(nf) / H1 (mV/Oe)");
else
    sprintf(label,"   P(nf) ( dBm )");

v_label(label);
labelx = xmin - (xmax - xmin) * .15;
labely = ymin;
v_move(labelx,labely);          v_textdir(0);
sprintf(label,"%9.2e",ymin);
v_label(label);
labelx = xmin - (xmax - xmin) * .15;
labely = ymax;
v_move(labelx,labely);
sprintf(label,"%9.2e",ymax);
v_label(label);

for (ii=1; ii<1000; ++ii)
    {
    fscanf(fileptr,"%d ",&samp_no);
    if (samp_no == ENDFILE)                 break;
    fscanf(fileptr,"%e %f %s %f %e %e %e"
      ,&ac_monV,&syn_amp,synampU,&syn_ph,&in_sig,&out_sig,&H1);

    logH1 = log10(H1);
    if (in_out == 1)
       yplot = (in_sig * cosfr + out_sig * sinfr) / H1;
    else if (in_out == 2)
       yplot = (-in_sig * sinfr + out_sig * cosfr) / H1;
    else
       {
       yplot = (in_sig * in_sig + out_sig * out_sig);
       yplot /= (775. * 775.);   /* Ref volt. for dBm = .775 Vrms */
       yplot = 10. * log10(yplot);
       }

    v_dot(logH1,yplot);
    }

fclose(fileptr);
getchar();          getchar();

v_initgraph();
v_scale(xmin,xmax,ymin,ymax);
labelx = xmin - (xmax - xmin) * 0.125;
labely = ymax;
for (ii=0; ii < textline; ++ii)
    {
    v_move(labelx,labely);
    v_label(line[ii]);
    labely -= ((ymax - ymin) * 0.06);
    }
getchar();              getchar();
clearviewport();        closegraph();
}
```

```
/*
**      keithley.c:  Routines which help control Keithley 197
**      DMM thru National Instrument GPIB-PCII.
**      Configured by Harry Lam, June 1990.
*/

#include <stdio.h>
#include "decl.h"
#define REAL float

/* Initializes the Keithley DMM to AUTO mode and return unit descriptor */
keith_in(intstrng)
char *intstrng;   /* intstrng contains the device name of the unit */
   {
   int ud;
   char dummy;

   ud = ibfind(intstrng);
   k_ready(ud); ibclr(ud);
   k_ready(ud); ibwrt(ud,"R0X",3);
   k_ready(ud);

   printf("\007");
   printf("Manually set the Keithley %3s as needed, then hit <CR>.",intstrng);
   scanf("%c",dummy);

   return(ud);
   }


/* Reads the signal on the meter, which is assumed to be in AUTO mode */
REAL keith_rd(uudd,unit,okay)
int uudd;
char unit[],okay; /* unit has to be at least 4 bytes long */
   {
   int ii,jj;
   char string[16],readbuf[50];
   REAL reading;

   for (jj=0; jj<5; ++jj)
      {
      k_ready(uudd);          k_rddone(uudd);
      ibrd(uudd,readbuf,17);

      for (ii=0; ii<15; ++ii)   string[ii] = readbuf[ii];
      string[15] = '\0';

      if (string[0] == 'O')
         {
         printf("\007\007\007\nDMM is overranged ");
         printf("; reset and then hit <CR>.");
         getchar();
         printf("Continuing ...\n");
         }
```

```
        else
            break;
        }

    sscanf(string,"%c%3s%11e",&okay,unit,&reading);
    k_rddone(uudd);
    return(reading);
    }

/* Make sure that the DMM is not busy */
k_ready(uudd)
int uudd;
    {
    char statbyte;
    int ii;

    ibwait(uudd,CMPL);
    for (ii=0; ii < 40; ++ii)
        {
        ibrsp(uudd,&statbyte);
        if (!(statbyte & 48))       break;   /* not error and not busy */
        if (ii == 39)
            {
            printf("\007\007\007");
            printf("Warning:  Keithley DMM has error or is always busy!\n");
            printf("DOUBLE CHECK if things are okay !!\n");
            }
        }
    return((int)statbyte);
    }

/* Make sure that the DMM is ready for another reading */
k_rddone(uudd)
int uudd;
    {
    char statbyte;
    int ii;

    ibwait(uudd,CMPL);
    for (ii=0; ii < 40; ++ii)
        {
        ibrsp(uudd,&statbyte);
        /* not error and reading done */
        if (!(statbyte & 32) && (statbyte & 8))        break;
        if (ii == 39)
            {
            printf("\007\007\007");
            printf("Warning:  Keithley DMM has error or");
            printf(" cannot get ready for another reading !\n");
            printf("DOUBLE CHECK if things are okay !!\n");
            }
        }
    return((int)statbyte);
    }
```

```
/*
**      par5209.c:  Routines which help controlling PAR 5209
**      lock-in amplifier thru National Instrument GPIB-PCII.
**
**      Written in Turbo C.  [cf. the function sleep() of Turbo C.]
**      Configured by Harry Lam, June 1990.
*/

#include <stdio.h>
#include <dos.h>
#include "decl.h"
#define REAL float

/* Reads the parameter settings of the lock-in */
read_set(uudd,dynres,freqmode,filtfreq,reffreq,linefilt,rolloff,filtfunc)
int uudd;          /* uudd is the unit descriptor */
char dynres[],freqmode[],linefilt[],rolloff[],filtfunc[];
REAL *filtfreq,*reffreq;
   {
   int code,ii;
   char readbuf[50];

   cmd_rdy(uudd);       ibwrt(uudd,"DR",2);
   data_rdy(uudd);      ibrd(uudd,readbuf,2);
   switch(readbuf[0])
        {
        case '0':
           sprintf(dynres,"HI STAB");   break;
        case '1':
           sprintf(dynres,"NORM");      break;
        case '2':
           sprintf(dynres,"HI RES");
        }

   cmd_rdy(uudd);       ibwrt(uudd,"F2F",3);
   data_rdy(uudd);      ibrd(uudd,readbuf,2);
   switch(readbuf[0])
        {
        case '0':
           sprintf(freqmode,"F mode");    break;
        case '1':
           sprintf(freqmode,"2F mode");
        }

   /* Read filter tuned frequency in hertz */
   cmd_rdy(uudd);       ibwrt(uudd,"FF",2);
   data_rdy(uudd);      ibrd(uudd,readbuf,5);
   *filtfreq = (REAL)(atoi(readbuf));

   data_rdy(uudd);      ibrd(uudd,readbuf,2);
   code = atoi(readbuf);

   if (code > 2)
        {
```

```
    for (ii=0; ii < (code-2); ++ii)        *filtfreq *= 10.;
    }
else if (code < 2)
    {
    for (ii=0; ii < (2-code); ++ii)        *filtfreq /= 10.;
    }

/* Read reference channel frequency in hertz */
cmd_rdy(uudd);        ibwrt(uudd,"FRQ",3);
data_rdy(uudd);       ibrd(uudd,readbuf,10);
readbuf[9] = '\0';    /* this statem't should be necessary */
sscanf(readbuf,"%9f",reffreq);
*reffreq /= 1000.;

cmd_rdy(uudd);        ibwrt(uudd,"LF",2);
data_rdy(uudd);       ibrd(uudd,readbuf,2);
switch(readbuf[0])
    {
    case '0':
       sprintf(linefilt,"OFF");                  break;
    case '1':
       sprintf(linefilt,"2F ON");                break;
    case '2':
       sprintf(linefilt,"F ON");                 break;
    case '3':
       sprintf(linefilt,"BOTH ON");
    }

cmd_rdy(uudd);        ibwrt(uudd,"XDB",3);
data_rdy(uudd);       ibrd(uudd,readbuf,2);
switch(readbuf[0])
    {
    case '0':
       sprintf(rolloff,"6 dB/octave");  break;
    case '1':
       sprintf(rolloff,"12 dB/octave");
    }

cmd_rdy(uudd);        ibwrt(uudd,"FLT",3);
data_rdy(uudd);       ibrd(uudd,readbuf,2);
switch(readbuf[0])
    {
    case '0':
       sprintf(filtfunc,"FLAT");                 break;
    case '1':
       sprintf(filtfunc,"NOTCH");                break;
    case '2':
       sprintf(filtfunc,"LOW-PASS"); break;
    case '3':
       sprintf(filtfunc,"BAND-PASS");
    }
 }

/* Reads the signal detected at present phase in milli-volts */
```

```
REAL sig_out(uudd,sencode)
int uudd,*sencode;
   {
   char statbyte,readbuf[50];
   int output,ii;
   REAL lu_sen(),sensit,signal;

   cmd_rdy(uudd);          ibwrt(uudd,"D2",2);
   data_rdy(uudd);         ibrd(uudd,readbuf,2);
   output = atoi(readbuf); /* atoi() converts only digit characters */

   if (output != 0 && output != 1 && output != 2)
        {
        printf("\007");
        printf("Put OUTPUT in SIGNAL mode, then hit <CR>.");
        getchar();
        }

   cmd_rdy(uudd);       ibwrt(uudd,"SEN",3);
   data_rdy(uudd);      ibrd(uudd,readbuf,3);
   *sencode = atoi(readbuf);

   for (ii=0; ii < 5; ++ii)
        {
        if (li_ovld(uudd) == 1)          /* Check overloading */
           {
           if ((*sencode) < 15)
                {
                ++(*sencode); lisetsen(uudd,*sencode);
                lkinwait(uudd); lkinwait(uudd);
                }
           else
                {
                printf("\007\007\007Signal or output ");
                printf("is overloaded!! Hit <CR> after reset.");
                getchar();
                }
           }

        /* Look-up table for sensitivity in millivolts */
        switch(*sencode)
           {
           case 0:          sensit = 0.0001;      break;
           case 1:          sensit = 0.0003;      break;
           case 2:          sensit = 0.001;       break;
           case 3:          sensit = 0.003;       break;
           case 4:          sensit = 0.01;        break;
           case 5:          sensit = 0.03;        break;
           case 6:          sensit = 0.1; break;
           case 7:          sensit = 0.3; break;
           case 8:          sensit = 1.;  break;
           case 9:          sensit = 3.;  break;
           case 10:      sensit = 10.;    break;
           case 11:      sensit = 30.;    break;
```

```
            case 12:        sensit = 100.;   break;
            case 13:        sensit = 300.;   break;
            case 14:        sensit = 1000.;  break;
            case 15:        sensit = 3000.;
            }

        cmd_rdy(uudd); ibwrt(uudd,"OUT",3);
        data_rdy(uudd); ibrd(uudd,readbuf,7);
        output = atoi(readbuf);

        if ( (output > -1000 && output < 1000) && (*sencode > 0)
                && (li_ovld(uudd) != 1) )
            {
            --(*sencode);                    lisetsen(uudd,*sencode);
            lkinwait(uudd);
            }
        else if ( (output < -10200 || output > 10200
                || li_ovld(uudd) == 1) && (*sencode < 15) )
            {
            ++(*sencode);                    lisetsen(uudd,*sencode);
            lkinwait(uudd);
            }
        else
            break;
        }

    signal = sensit * ((REAL)output) / 10000.;
    return(signal);
    }

/* Set lock-in sensitivity according to input code number */
lisetsen(uudd,code)
int uudd,code;
    {
    char cmd[10],statbyte;

    sprintf(cmd,"SEN %2d",code);
    cmd_rdy(uudd);      ibwrt(uudd,cmd,6);

    cmd_rdy(uudd);
    }

/* Set lock-in sensitivity according to input voltage (in mV) */
set_sen(uudd,sensit)
int uudd;
REAL sensit;
    {
    int code;

    if      (sensit < 0.00015)                          code = 0;
    else if (sensit >= 0.00015 && sensit < 0.00035)     code = 1;
    else if (sensit >= 0.00035 && sensit < 0.0015)      code = 2;
    else if (sensit >= 0.0015 && sensit < 0.0035)  code = 3;
    else if (sensit >= 0.0035 && sensit < 0.015)   code = 4;
```

```
    else if (sensit >= 0.015 && sensit < 0.035)          code = 5;
    else if (sensit >= 0.035 && sensit < 0.15)           code = 6;
    else if (sensit >= 0.15 && sensit < 0.35)            code = 7;
    else if (sensit >= 0.35 && sensit < 1.5)             code = 8;
    else if (sensit >= 1.5 && sensit < 3.5)              code = 9;
    else if (sensit >= 3.5 && sensit < 15.)              code = 10;
    else if (sensit >= 15. && sensit < 35.)              code = 11;
    else if (sensit >= 35. && sensit < 150.)             code = 12;
    else if (sensit >= 150. && sensit < 350.)            code = 13;
    else if (sensit >= 350. && sensit < 1500.)           code = 14;
    else if (sensit >= 1500.)                            code = 15;

    lisetsen(uudd,code);
    }

/*  Reads current time-constant. */
REAL rd_timec(uudd)
int uudd;
    {
    REAL timec;
    int n1;
    char buf1[5];

    cmd_rdy(uudd);        ibwrt(uudd,"XTC",3);
    data_rdy(uudd);
    ibrd(uudd,buf1,3);                          n1 = atoi(buf1);

    /* Look-up table for time constant in seconds */
    switch(n1)
         {
         case 0:    timec = .001;  break;
         case 1:    timec = .003;  break;
         case 2:    timec = .010;  break;
         case 3:    timec = .030;  break;
         case 4:     timec = .100; break;
         case 5:     timec = .300; break;
         case 6:    timec = 1.00;  break;
         case 7:    timec = 3.00;  break;
         case 8:    timec = 10.0;  break;
         case 9:    timec = 30.0;  break;
         case 10:   timec = 100.;  break;
         case 11:   timec = 300.;  break;
         case 12:   timec = 1000.;          break;
         case 13:   timec = 3000.;
         }

     return(timec);
     }

/*  Step up or down the time-constant by 'updown' step(s).  */
stptimec(uudd,updown)
int uudd,updown;
    {
    int n1;
```

```
   char buf1[8];

   cmd_rdy(uudd);        ibwrt(uudd,"XTC",3);
   data_rdy(uudd);       ibrd(uudd,buf1,3);

   n1 = atoi(buf1);      n1 + = updown;

   if (n1 < 0)           n1 = 0;
   else if (n1 > 13)     n1 = 13;

   sprintf(buf1,"XTC %2d",n1);
   cmd_rdy(uudd);        ibwrt(uudd,buf1,6);
   cmd_rdy(uudd);
   }

/* Enable the computer to wait an appropriate time after a new
** sensitivity level is set before reading the signal output
*/

lkinwait(uudd)
int uudd;
   {
   REAL timec,rd_timec();
   unsigned second = 2;
   char buf1[5];
   int n1;

   timec = rd_timec(uudd);

   cmd_rdy(uudd);        ibwrt(uudd,"XDB",3);
   data_rdy(uudd);
   ibrd(uudd,buf1,2);                    n1 = atoi(buf1);

   switch(n1)
        {
        case 0:   timec *= 4.5;          break;    /* 6 dB/octave */
        case 1:    timec *= 7.2;                   /* 12 dB/octave */
        }

   if (timec > 1.)     second = ((int)timec) + 1;

   sleep(second);
   }

/*  Reads current phase; returns phase in degrees */
REAL rd_phase(uudd)
int uudd:
   {
   REAL phase;
   int n1;
   char buf1[5],buf2[10];

   cmd_rdy(uudd);        ibwrt(uudd,"P",1);
   data_rdy(uudd);
```

```
   ibrd(uudd,buf1,2);              n1 = atoi(buf1);
   data_rdy(uudd);
   ibrd(uudd,buf2,7);                    buf2[6] = '\0';
   sscanf(buf2,"%f",&phase);             phase /= 1000.;

   switch(n1)
        {
        case 0:   phase += 0.;           break;
        case 1:  phase += 90.;           break;
        case 2:  phase += 180.;         break;
        case 3:  phase += 270.;
        }

   if (phase > 180.)               phase -= 360.;

   return(phase);
   }

/* Shifts the current by a specified number of quadrants, pos. or neg. */
shftquad(uudd,quad)
int quad,uudd;              /* quad: number of quadrants to be shifted */
   {
   int n1;
   long n2;    /* Warning: watch the size of "long" for the compiler */
   char statbyte,buf1[5],buf2[7],command[15];

   cmd_rdy(uudd);        ibwrt(uudd,"P",1);
   data_rdy(uudd);       ibrd(uudd,buf1,2);    n1 = atoi(buf1);
   data_rdy(uudd);    ibrd(uudd,buf2,7);    buf2[6] = '\0';
   sscanf(buf2,"%ld",&n2);

   n1 -= quad;                    n1 %= 4;
   if (n1 < 0)             n1 += 4;

   sprintf(command,"P %d,%6ld",n1,n2);

   cmd_rdy(uudd);    ibwrt(uudd,command,10);     cmd_rdy(uudd);
   }

/* Make sure that lock-in is ready for the next command */
cmd_rdy(uudd)
int uudd;
   {
   char statbyte = 0;
   int ii;
   extern int board_ud;

   ibwait(uudd,CMPL);
   for (ii=0; ii < 40; ++ii)
        {
        ibrsp(uudd,&statbyte);

        if (statbyte & 1)          break;
        if (ii == 39)
```

```
            {
            printf("\007\007\007");
            printf("**Warning: SOMEHOW LOCK-IN DID NOT INDICATE ");
            printf("COMMAND DONE, CHECK DATA !!\n");
            printf("Lock-in statbyte = %d\n",(int)statbyte);
            }
        }
    return((int)statbyte);
    }

/* Make sure that lock-in's output is ready to be sent */
data_rdy(uudd)
int uudd;
    {
    char statbyte = 0;
    int ii;
    extern int board_ud;

    ibwait(uudd,CMPL);
    for (ii=0; ii < 40; ++ii)
        {
        ibrsp(uudd,&statbyte);

        if (statbyte & 128)        break;
        if (ii == 39)
            {
            printf("\007\007\007");
            printf("**Warning: SOMEHOW LOCK-IN DID NOT INDICATE ");
            printf("OUTPUT READY, CHECK DATA !!\n");
            printf("Lock-in statbyte = %d\n",(int)statbyte);
            }
        }
    return((int)statbyte);
    }

/* Check if lock-in is overload */
li_ovld(uudd)              /* return 1 if overload, else 0 */
int uudd;
    {
    char statbyte;
    int ii;

    ibwait(uudd,CMPL);          ibrsp(uudd,&statbyte);

    if (statbyte & 16)              ii = 1;
    else                    ii = 0;

    return(ii);
    }
```

```
/*
**      hp3325.c:  Routines which help control HP 3325
**      synthesizers thru National Instrument GPIB-PCII.
**      Configured by Harry Lam, June 1990.
*/

#include <stdio.h>
#include "decl.h"
#define REAL float

char buffer[50];

/* Read the phase parameter of the synthesizer, in degrees */
/* uudd is the unit descriptor */
REAL synphase(uudd)
int uudd;
   {
   REAL phase;

   syn_rdy(uudd);          ibwrt(uudd,"IPH",3);
   syn_rdy(uudd);          ibrd(uudd,buffer,18);
   buffer[16] = '\0';
   sscanf(buffer,"%*2c%12f%*2c",&phase);
   return(phase);
   }

/* Set the peak-to-peak amplitude of the synthesizer. */
synsetam(uudd,amplit)
int uudd;
REAL amplit;  /* in volts */
   {
   REAL amplitMV;
   char buf[12];
   if (amplit >= 1.0)
        sprintf(buf,"AM%7.4fVO",amplit);
   else
        {
        amplitMV = amplit * 1000.;
        sprintf(buf,"AM%7.3fMV",amplitMV);
        }
   syn_rdy(uudd);          ibwrt(uudd,buf,11);
   }

/* Read the amplitude parameter of the synthesizer. */
REAL synampl(uudd,unit)
int uudd;
char unit[];
   {
   REAL amplit;
   char delim[3];

   syn_rdy(uudd);          ibwrt(uudd,"IAM",3);
   syn_rdy(uudd);          ibrd(uudd,buffer,18);
   buffer[16] = '\0';
```

```
   sscanf(buffer,"%*2c%12f%2s",&amplit,delim);
   switch(delim[0])
         {
         case 'V':
            if (delim[1] = = 'O')    sprintf(unit,"Vpp");
            else                     sprintf(unit,"Vrms");
            break;
         case 'M':
            if (delim[1] = = 'V')    sprintf(unit,"mVpp");
            else                     sprintf(unit,"mVrms");
            break;
         case 'D':
            sprintf(unit,"dBm");
         }
   return(amplit);
   }

/* Reads the frequency parameter of the synthesizer */
REAL synfreq(uudd,unit)
int uudd;
char unit[];
   {
   char delim[3];
   REAL freq;

   syn_rdy(uudd);          ibwrt(uudd,"IFR",3);
   syn_rdy(uudd);          ibrd(uudd,buffer,18);
   buffer[16] = '\0';
   sscanf(buffer,"%*2c%12f%2s",&freq,delim);
   switch(delim[0])
         {
         case 'H':  sprintf(unit,"Hz");    break;
         case 'K':  sprintf(unit,"kHz");   break;
         case 'M':  sprintf(unit,"MHz");
         }
   return(freq);
   }

/* Read the miscellaneous parameters of the synthesizer. */
REAL synparam(uudd,osunit,func,hivolt)
int uudd;
char osunit[],func[],hivolt[];
   {
   REAL offset;
   int code;
   char delim[3];

   syn_rdy(uudd);          ibwrt(uudd,"IHV",3);
   syn_rdy(uudd);          ibrd(uudd,buffer,5);
   switch(buffer[2])
         {
         case '0':   sprintf(hivolt,"hivolt off");    break;
         case '1':   sprintf(hivolt,"hivolt on");
         }
```

```
    syn_rdy(uudd);          ibwrt(uudd,"IFU",3);
    syn_rdy(uudd);          ibrd(uudd,buffer,5);
    switch(buffer[2])
          {
          case '0':   sprintf(func,"DC only");              break;
          case '1':   sprintf(func,"Sine");         break;
          case '2':   sprintf(func,"Square");       break;
          case '3':   sprintf(func,"Triangle");             break;
          case '4':   sprintf(func,"Pos. ramp");            break;
          case '5':   sprintf(func,"Neg. ramp");
          }

    syn_rdy(uudd);          ibwrt(uudd,"IOF",3);
    syn_rdy(uudd);          ibrd(uudd,buffer,18);
    buffer[16] = '\0';
    sscanf(buffer,"%*2c%12f%2s",&offset,delim);
    switch(delim[0])
          {
          case 'V':   sprintf(osunit,"Vdc");      break;
          case 'M':   sprintf(osunit,"mVdc");
          }
    return(offset);
    }

/* Make sure the synthesizer is NOT busy and is ready */
syn_rdy(uudd)
int uudd;
    {
    char statbyte = 128;
    int ii;

     * See pp. 3-24 of HP3325A manual */
    ibwait(uudd,CMPL);

    for (ii = 0; ii < 40; --ii)
          {
          ibrsp(uudd,&statbyte);

          if ( !(statbyte & 128))               break;
          if (ii == 39)
              {
              printf("Warning:  SOMEHOW A SYNTHESIZER IS ALWAYS");
              printf(" BUSY;  CHECK DATA !!\n\007\007\007");
              }
          }
     return((int)statbyte);
     }
```

```
/*
**      hp3585sa.c:  Routines which help control HP 3585A
**      spectrum analyzer thru National Instrument GPIB-PCII.
**      Configured by Harry Lam, August 1990.
*/

#include <stdio.h>
#include "decl.h"
#define REAL float

/* Read the alphanumerics of the spectrum analyzers.  Correct
   memory sizes for the different string variables must be
   assigned by the calling program:  ref[15], mkrfrq[26],
   dbdiv[10], range[16], mkramp[15], ctrfrq[23], span[21],
   rbw[12], vbw[12], st[13].
   Will return programming code's "data" value for present
   RANGE setting; cf. manual's page 3-9-17.
*/
sa_alnum(uudd,ref,mkrfrq,dbdiv,range,mkramp,ctrfrq,span,rbw,vbw,st)
int uudd;          /* uudd is the unit descriptor */
char ref[],mkrfrq[],dbdiv[],range[],mkramp[];
char ctrfrq[],span[],rbw[],vbw[],st[];
    {
    char dump;
    int rngindex;
    REAL rnglvl;

    ibwrt(uudd,"D7T4\012",5);      ibwait(uudd,CMPL);

    ibrd(uudd,ref,15);             ibrd(uudd,&dump,1);
    ibrd(uudd,mkrfrq,26);          ibrd(uudd,&dump,1);
    ibrd(uudd,dbdiv,10);   ibrd(uudd,&dump,1);
    ibrd(uudd,range,16);   ibrd(uudd,&dump,1);
    ibrd(uudd,mkramp,15);          ibrd(uudd,&dump,1);
    ibrd(uudd,ctrfrq,23);  ibrd(uudd,&dump,1);
    ibrd(uudd,span,21);            ibrd(uudd,&dump,1);
    ibrd(uudd,rbw,12);             ibrd(uudd,&dump,1);
    ibrd(uudd,vbw,12);             ibrd(uudd,&dump,1);
    ibrd(uudd,st,13);              ibrd(uudd,&dump,1);

    ref[14] = mkrfrq[25] = dbdiv[9] = range[15] = mkramp[14] = '\0';
    ctrfrq[22] = span[20] = rbw[11] = vbw[11] = st[12] = '\0';

    sscanf(range,"%*5s%f",&rnglvl);

    if (rnglvl >= -26. && rnglvl <= -24.)          rngindex = 1;
    else if (rnglvl >= -21. && rnglvl <= -19.)  rngindex = 2;
    else if (rnglvl >= -16. && rnglvl <= -14.)  rngindex = 3;
    else if (rnglvl >= -11. && rnglvl <= -9.)   rngindex = 4;
    else if (rnglvl >= -6.  && rnglvl <= -4.)   rngindex = 5;
    else if (rnglvl >= -1.  && rnglvl <= 1.)       rngindex = 6;
    else if (rnglvl >= 4.   && rnglvl <= 6.)       rngindex = 7;
    else if (rnglvl >= 9.   && rnglvl <= 11.)   rngindex = 8;
    else if (rnglvl >= 14.  && rnglvl <= 16.)   rngindex = 9;
```

```
    else if (rnglvl > = 19.  && rnglvl < = 21.)   rngindex = 10;
    else if (rnglvl > = 24.  && rnglvl < = 26.)   rngindex = 11;
    else if (rnglvl > = 29.  && rnglvl < = 31.)   rngindex = 12;

    return(rngindex);
    }

/* Reads the marker's amplitude. */
REAL sa_mkamp(uudd)
int uudd;
    {
    char statbyte = 0,buffer[20];
    REAL amplit;
    int ii;

    ibwrt(uudd,"D1T5\012",5);      ibwait(uudd,CMPL);

    for (ii=0; ii < 10000; + +ii)
         {
         ibrsp(uudd,&statbyte);

         if ( (statbyte & 64) && (statbyte & 2) )    break;

         if (ii = = 9999)
            {
            printf("\007\007\007**WARNING:  ");
            printf("Somehow the spectrum analyzer could ");
            printf("not output marker amplitude!!\n");
            printf("Spectrum analyzer statbyte = %d\n",(int)statbyte);
            }
         }

    ibrd(uudd,buffer,13);  buffer[11] = '\0';

    sscanf(buffer,"%e",&amplit);    return(amplit);
    }

/*  Converts programming code's RANGE "data" to actual range level
    according to the table on manual's page 3-9-17.
*/
REAL sa_Rconv(rngindex)
int rngindex;
    {
    REAL rnglvl;

    switch(rngindex)
         {
         case 1:    rnglvl = -25.;   break;
         case 2:    rnglvl = -20.;   break;
         case 3:    rnglvl = -15.;   break;
         case 4:     rnglvl = -10.;  break;
         case 5:    rnglvl =  -5.;   break;
         case 6:    rnglvl =   0.;   break;
         case 7:    rnglvl =   5.;   break;
```

```
        case 8:   rnglvl =  10.;  break;
        case 9:   rnglvl =  15.;          break;
        case 10:  rnglvl =  20.;  break;
        case 11:  rnglvl =  25.;  break;
        case 12:  rnglvl =  30.;
        }

   return(rnglvl);
   }

/* Set range level according to input "rngindex" value */
sasetrng(uudd,rngindex)
int uudd,rngindex;
   {
   char buf[5];

   if (rngindex < 10 && rngindex > 0)
        sprintf(buf,"R0%1d\012",rngindex);
   else if (rngindex > = 10 && rngindex < = 12)
        sprintf(buf,"R%2d\012",rngindex);
   else if (rngindex < = 0)
        sprintf(buf,"R01\012");
   else
        sprintf(buf,"R12\012");

   ibwrt(uudd,buf,4);              ibwait(uudd,CMPL);
   }

/* Set manual frequency to input value "freq" in kHz */
sa_manF(uudd,freq)
int uudd;
REAL freq;
   {
   char buf[20];

   sprintf(buf,"S3%10.4fKZ\012",freq);
   ibwrt(uudd,buf,15);             ibwait(uudd,CMPL);
   }

/* Switch input port: 1 -> 1 Mohm; 2 -> 50 ohms; 3 -> 75 ohms. */
sa_port(uudd,code)
int uudd,code;
   {
   char buf[3];

   sprintf(buf,"I%d\012",code);
   ibwrt(uudd,buf,3);              ibwait(uudd,CMPL);
   }
```

```
/*  hpgraph.c:  Graphics driver for HP7225B plotter.
**      The computer is assumed to communicate with
**      the plotter thru National Instrument GPIB-PCII.
**      Configured by Harry Lam, June 1990, based on a
**      similar program written by Jim Crutchfield for
**      the Sun workstations.
*/

#include <stdio.h>
#include <math.h>
#include "decl.h"
#define REAL float
#define PI 3.141592654
#define LOG2  0.301029996
#define LOG3  0.477121255
#define LOG4  0.602059991
#define LOG5  0.698970004
#define LOG6  0.77815125
#define LOG7  0.84509804
#define LOG8  0.903089987
#define LOG9  0.954242509

char command[100];
int uudd;
int gr_width,gr_hght;
int gr_1x,gr_2x,gr_1y,gr_2y;
int p1x,p1y,p2x,p2y;
REAL bx0,bx1,by0,by1;
REAL xscale,yscale;

initgraph(intstrng)   /* Unit descriptor will be returned */
/* intstrng contains the device name of the plotter */
char *intstrng;
   {
   /* Default plot/write area */
   gr_1x = 328;   gr_2x = 10328;   gr_1y = 300;   gr_2y = 8000;
   gr_width = gr_2x - gr_1x;      gr_hght = gr_2y - gr_1y;
   /* Defining default plotting window */
   p1x = 1500;     p2x = 9500;     p1y = 1500;     p2y = 7500;
   /* Default scaling */
   bx0 = 1500.;   bx1 = 9500.;   by0 = 1500.;   by1 = 7500.;
   xscale = yscale = 1.;

   uudd = ibfind(intstrng);
   ibclr(uudd);
   ibwrt(uudd,"IN;",3);
   return(uudd);
   }

char clear()   /* Ask if new paper is in place & plotter on and ready. */
   {
   printf("Hit key 'c' when plotter is ready with fresh paper\n");
   while ((getchar()) != 'c');
   printf("Continuing\n");
```

```
    }

/* Scale the device units with the user's units */
scale(x0,x1,y0,y1)
REAL x0,x1,y0,y1;
    {
    bx0 = x0;    bx1 = x1;    by0 = y0;    by1 = y1;
    }

move(x,y)
REAL x,y;
    {
    int xtodev(), ytodev(),ii,ix,iy;  /* The 2 functions convert user units
                                         to plotter units. */

    ix = xtodev(x);          iy = ytodev(y);
    sprintf(command,"PA%5d,%5d;",ix,iy);
/*    if (gr_1x < = ix && gr_2x > = ix && gr_1y < = iy && gr_2y > = iy)   */
        for (ii=0; ii<14; ++ii)      ibwrt(uudd,command+ii,1);
    }

border()
    {
    move(bx0,by0);          pendown();
    move(bx1,by0);          move(bx1,by1);      move(bx0,by1);
    move(bx0,by0);          penup();
    }

axes(xorigin,yorigin,xtic,ytic,upright,downleft)
REAL xorigin,yorigin,xtic,ytic;
int upright,downleft;
    {
    int ii;
    REAL t;

    if (upright < 0 || upright > 100)        upright = 5;
    if (downleft < 0 || downleft >100)       downleft = 5;

    sprintf(command,"TL%3d,%3d;",upright,downleft);
    for (ii=0; ii<10; ++ii)  ibwrt(uudd,command+ii,1);

    if (bx0 < = xorigin && xorigin < = bx1)
        {
        if (by0 < = yorigin && yorigin < = by1)
            {
            move(xorigin,yorigin);   pendown();
            for (t=yorigin; t < = by1; t += ytic)
                {
                move(xorigin,t);   ibwrt(uudd,"YT;",3);
                }
            penup();
            move(xorigin,yorigin);   pendown();
            for (t = yorigin-ytic; t > = by0; t -= ytic)
                {
```

```
                move(xorigin,t);   ibwrt(uudd,"YT;",3);
                }
            penup();
            move(xorigin,yorigin);   pendown();
            for (t=xorigin; t <= bx1; t += xtic)
                {
                move(t,yorigin);   ibwrt(uudd,"XT;",3);
                }
            penup();
            move(xorigin,yorigin);   pendown();
            for (t = xorigin-xtic; t >= bx0; t -= xtic)
                {
                move(t,yorigin);   ibwrt(uudd,"XT;",3);
                }
            penup();
            }
        }
    }

/* To draw the x-axes for semi-log graph. */
log_xaxes(xmin,xmax,ymin,ymax,updown)
REAL xmin,xmax,ymin,ymax;
int updown;
    {
    int ii,jj;
    REAL xplot,xplot0,yplot;

    if (updown < 0 || updown > 100)     updown = 5;

    for (jj=0; jj<2; ++jj)
        {
        if (jj == 0)
            {
            yplot = ymin;    sprintf(command,"TL%3d,0;",updown);
            }
        else
            {
            yplot = ymax;    sprintf(command,"TL0,%3d;",updown);
            }

        for (ii=0; ii<8; ++ii)     ibwrt(uudd,command+ii,1);

        xplot0 = xmin;             move(xplot0,yplot);
        pendown();

        for (ii=0; ii<100; ++ii)
            {
            xplot = xplot0 + LOG2;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG3;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG4;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG5;
```

```
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG6;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG7;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG8;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + LOG9;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot = xplot0 + 1.;
            move(xplot,yplot);              ibwrt(uudd,"XT;",3);
            xplot0 = xplot;
            if (xplot0 >= xmax)             break;
            }
        penup();
        }
    }

xtodev(x)       /* Convert x-coord from usr's scale to plotter's scale */
REAL x;
    {
    int ilocx;

    ilocx = (int)( (x - bx0) / (bx1 - bx0) * (p2x - p1x) );
    ilocx += p1x;
    return(ilocx);
    }

ytodev(y)       /* Convert y-coord from usr's scale to plotter's scale */
REAL y;
    {
    int ilocy;

    ilocy = (int)( (y - by0) / (by1 - by0) * (p2y - p1y) );
    ilocy += p1y;
    return(ilocy);
    }

charsize(xdim,ydim)     /* character size, in cm */
REAL xdim,ydim;
    {
    int ii;
    if (xdim < -10. || xdim > 10. || ydim < -10. || ydim > 10.)
        {
        xdim = .2;      ydim = .4;
        }
    sprintf(command,"SI%6.2f,%6.2f;",xdim,ydim);
    for (ii=0; ii<16; ++ii)         ibwrt(uudd,command+ii,1);
    }

labeldir(angle)  /* angle in degrees; 0 = horiz. */
REAL angle;
    {
    REAL dgtord = PI / 180.,radian,run,rise;
```

```
    int ii;

    radian = angle * dgtord;
    run = cos(radian);              rise = sin(radian);
    if (angle = = 0.)
          {
          sprintf(command,"DI:");
          for (ii=0; ii<3; + +ii)    ibwrt(uudd,command+ii,1);
          }
    else
          {
          sprintf(command,"DI%7.4f,%7.4f;",run,rise);
          for (ii=0; ii<18; + +ii)   ibwrt(uudd,command+ii,1);
          }
    }

label(string)
char *string;
    {
    int length;

    length = strlen(string);
    ibwrt(uudd,"LB",2);    ibwrt(uudd,string,length);
    ibwrt(uudd,"\003",1);
    }

circleat(x,y,radius,chordang)
    /* x,y in usr's units; radius in scaled x units */
REAL x,y,radius;
    /* chordang in integral degrees */
int chordang;
    {
    int r,jj,angle,ix,iy,iccx,iccy;
    REAL relx,rely,relangcc,relang;

    angle = chordang % 360;
    relangcc = ((REAL)angle) / 180. * PI;
    r = (int)(radius / (bx1 - bx0) * (p2x - p1x));
    ix = xtodev(x);          iy = ytodev(y);

    for (relang = 0.; relang < = (2.1*PI); relang + = relangcc)
          {
          iccx = ix + (int)((REAL)r * cos(relang));
          iccy = iy + (int)((REAL)r * sin(relang));
          sprintf(command,"PA%6d,%6d;",iccx,iccy);
          for (jj=0; jj<16; + +jj)   ibwrt(uudd,command+jj,1);

          if (relang = = 0.)          pendown();
          }
     penup();
     }

 dotat(x,y)          /*  x, y in usr's scale */
 REAL x,y;
```

```
    {
    move(x,y);              pendown();      penup();
    }

penup()             { ibwrt(uudd,"PU;",3); }

pendown()   { ibwrt(uudd,"PD;",3); }
```

```
/*
**      v_graph.c:  Turbo C graphics driver for the video.
**      Configured by Harry Lam, June 1990.
*/

#include <graphics.h>
#include <stdio.h>
#include <conio.h>
#include <process.h>
#define VPCLIP    1    /* Clip output when beyond viewport */
#define REAL  float
#define LOG2  0.301029996
#define LOG3  0.477121255
#define LOG4  0.602059991
#define LOG5  0.698970004
#define LOG6  0.77815125
#define LOG7  0.84509804
#define LOG8  0.903089987
#define LOG9  0.954242509

int gr_1x,gr_2x,gr_1y,gr_2y;
int p1x,p2x,p1y,p2y;
int font,directn,chsize,v_just,h_just;
int tmarkx,tmarky;

REAL bx0,bx1,by0,by1;

v_initgraph()
   {
   int g_driver,g_mode,g_error;

   /* Default plot/write graphics area */
   gr_1x = 0;   gr_2x = 639;   gr_1y = 15/* 5*/;   gr_2y = 349/* 479*/;
   /* Defining default plotting window */
   p1x = 100;   p2x = 600;   p1y = 25/*15*/;   p2y = 275 /* 379*/;
   /* Default scaling */
   bx0 = 100.;  bx1 = 600.;  by0 = 25./*15.*/;  by1 = 275. /*379*/;
   /* Default text setting */
   font = 0;   directn = HORIZ_DIR;   chsize = 1;
   h_just = LEFT_TEXT;                v_just = BOTTOM_TEXT;
   /* Default tick mark size in graphics scale */
   tmarkx = 3;   tmarky = 3;

   g_driver = DETECT;

/*   g_driver = VGA;  g_mode = VGAHI; */

   initgraph(&g_driver,&g_mode,"");

   g_driver = VGA;     g_mode = VGAHI;

   g_error = graphresult();
   if (g_error < 0)
        printf("Initgraph error: %s\n",grapherrormsg(g_error));
```

```
    setviewport(gr_1x,gr_1y,gr_2x,gr_2y,VPCLIP);
    clearviewport();
    rectangle(0,0,gr_2x-gr_1x,gr_2y-gr_1y);
    settextstyle(font,directn,chsize);
    settextjustify(h_just,v_just);
    }

/* Scale the video's units to the user's units. */
v_scale(x0,x1,y0,y1)
REAL x0,x1,y0,y1;
    {
    bx0 = x0;    bx1 = x1;    by0 = y0;    by1 = y1;
    }

v_move(x,y)    /* Move the current position to (x,y) in usr's scale */
REAL x,y;
    {
    int v_xtodev(),v_ytodev(),ix,iy;

    ix = v_xtodev(x);       iy = v_ytodev(y);
    moveto(ix,iy);
    }

v_lineto(x,y)  /* Draw a line from CP to (x,y) in usr's scale, */
REAL x,y;      /* and move CP to (x,y).                        */
    {
    int v_xtodev(),v_ytodev(),ix,iy;

    ix = v_xtodev(x);       iy = v_ytodev(y);
    lineto(ix,iy);
    }

v_border()
    {
    rectangle(p1x,p1y,p2x,p2y);
    }

v_axes(xorigin,yorigin,xtic,ytic,updown,leftrght)
REAL xorigin,yorigin,xtic,ytic;
int updown,leftrght;
    {
    int dx,dy;
    REAL t;

    dx = tmarkx * leftrght;          dy = tmarky * updown;
    if (bx0 < = xorigin && xorigin < = bx1)
        {
        if (by0 < = yorigin && yorigin < = by1)
            {
            v_move(xorigin,yorigin);
            for (t=yorigin; t < = by1; t + = ytic)
                {
                v_lineto(xorigin,t);    linerel(dx,0);
                linerel(-2*dx,0); v_move(xorigin,t);
```

```
            }
        v_move(xorigin,yorigin);
        for (t = yorigin - ytic; t >= by0; t -= ytic)
            {
            v_lineto(xorigin,t);        linerel(dx,0);
            linerel(-2*dx,0); v_move(xorigin,t);
            }
        v_move(xorigin,yorigin);
        for (t=xorigin; t <= bx1; t += xtic)
            {
            v_lineto(t,yorigin);        linerel(0,dy);
            linerel(0,-2*dy); v_move(t,yorigin);
            }
        v_move(xorigin,yorigin);
        for (t = xorigin-xtic; t >= bx0; t -= xtic)
            {
            v_lineto(t,yorigin);        linerel(0,dy);
            linerel(0,-2*dy); v_move(t,yorigin);
            }
        }
    }
}

/* To draw the x-axes for semi-log graph. */
v_logxaxes(xmin,xmax,ymin,ymax,updown)
REAL xmin,xmax,ymin,ymax;
int updown;
    {
    int dy1,dy2,ii,jj;
    REAL xplot0,xplot,yplot;

    dy1 = tmarky * updown;

    for (jj=0; jj<2; ++jj)
        {
        if (jj == 0)
            {
            yplot = ymin;       dy2 = -dy1;
            }
        else
            {
            yplot = ymax;       dy2 = dy1;
            }

        xplot0 = xmin;          v_move(xplot0,yplot);

        for (ii=0; ii<100; ++ii)
            {
            xplot = xplot0 + LOG2;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG3;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG4;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG5;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG6;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG7;      TICKS(xplot,yplot,dy2);
```

```
            xplot = xplot0 + LOG8;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + LOG9;      TICKS(xplot,yplot,dy2);
            xplot = xplot0 + 1.;  TICKS(xplot,yplot,dy2);
            xplot0 = xplot;
            if (xplot0 >= xmax)  break;
            }
        }
    }

TICKS(xplot,yplot,dy)
REAL xplot,yplot;
int dy;
    {
    v_lineto(xplot,yplot);  v_move(xplot,yplot);
    linerel(0,dy);          v_move(xplot,yplot);
    }

v_xtodev(x)    /* Convert x-coord from usr's scale to window's scale */
REAL x;
    {
    int ilocx;

    ilocx = (int)( (x - bx0) / (bx1 - bx0) * (p2x - p1x) );
    ilocx += p1x;
    return(ilocx);
    }

v_ytodev(y)    /* Convert y-coord from usr's scale to window's scale */
REAL y;
    {
    int ilocy;

    ilocy = (int)( (y - by0) / (by1 - by0) * (p1y - p2y) );
    ilocy += p2y;
    return(ilocy);
    }

v_charsize(cc)  /* chsize: magnification factor of default char size */
int cc;
    {
    chsize = cc;
    if (chsize == 0 || chsize > 10)        chsize = 1;
    settextstyle(font,directn,chsize);
    }

v_textdir(cc)  /* directn: 1 means vertical, else horizontal */
int cc;
    {
    directn = cc;
    if (directn != 1 )    directn = 0;
    settextstyle(font,directn,chsize);
    }

v_label(string)
```

```
char *string;
   { outtext(string); }

v_circle(x,y,radius)
REAL x,y,radius;          /* x,y in usr's scale; radius in x-scale */
   {
   int ix,iy,v_xtodev(),v_ytodev(),iradius;
   REAL dummy;

   ix = v_xtodev(x);      iy = v_ytodev(y);
   dummy = x + radius;

   iradius = v_xtodev(dummy);          iradius -= ix;
   circle(ix,iy,iradius);
   }

v_dot(x,y)          /* x, y in usr's scale */
REAL x,y;
   {
   int ix,iy,v_xtodev(),v_ytodev(),ii,jj;

   ix = v_xtodev(x);      iy = v_ytodev(y);

/*    for (ii = -1; ii <= 1; ++ii)
         {
         for (jj = -1; jj <= 1; ++jj)          putpixel(ix+ii,iy+jj,15);
         }  */

   putpixel(ix,iy,15);
   }

v_aspect()
   {
   int xasp,yasp;

   getaspectratio(&xasp,&yasp);
   setaspectratio(xasp,xasp);
   }
```

END

DATE FILMED

01/07/92