A NEW AC – RADIO FREQUENCY HEATING CALORIMETRY TECHNIQUE FOR COMPLEX FLUIDS

by

Saimir Barjami

Saimir Barjami

A Dissertation

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Doctor of Philosophy

in

Physics Department

May 2005

APPROVED:

Associate Professor Germano S. Iannacchione, Advisor

Germano Iannacchione

Professor Padmanabhan K. Aravind

P K Aravind

Assistant Professor Ranjan Mukhopadhyay

Clark University

Ranjan Mukhopadhyay

Table of Contents

Chapter 3

Chapter 4

Chapter 5

Chapter 6

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to all the people who have helped to make this work possible.

I would like to thank my advisor, Dr. Germano S. Iannacchione, for his invaluable guidance and encouragement throughout the course of this work. The work could not possibly have been completed without his assistance, patience and time.

I would like to thank all the collaborators directly affecting this work: Aleksander Roshi, Dipti Sharma, Marco Caggioni, Tommaso Bellini, Francesco Mantegazza.

The greatest thanks I give are to **my family**. It is their love, help and sacrifices that made what I have achieved possible. To my mother **Hajrie** and to my father **Barjam**.

Dear Mom, it was Your love, Your prayers, Your sacrifices that made this and every achievement in my life possible. There can be no way I can possibly repay You. I therefore proudly dedicate this work to You.

I would like to give special thanks to my brother **Jako Barjami** for his tremendous support, help and understanding during all the years of this work. Thank you brother, and God bless you. I would like to thank my brothers and sisters: **Bukurije**, **Muhamet**, **Gezim**, **Jako**, **Mane**, **Edlira** and **Krenar Barjami** for their love, help, support and

sacrifices throughout the years of my life. You are the best brothers and sisters a person can wish for. Thank you all and God bless you.

I would like to thank God for blessing me with this achievement.

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Liquid crystals display properties intermediate between those of an isotropic liquid and a crystalline solid phase. Traditionally for a long time it was believed that there are only three states of matter: gases, liquids and solids. Reinitzer [1] and Lehmann [2] were the first ones to observe a liquid crystal behavior. A huge number of well known compounds are liquid crystals, i.e. lecithin, DNA, cellulose, cholesterol esters, gangliosids, paraffins, and graphite. Liquid crystals exhibit several intermediate phases between a crystalline solid characterized by translational and orientational long range order and a true liquid (isotropic fluid) which has perfect symmetry. Transitions to these intermediate states can occur by purely thermal processes (thermotropic mesomorphism) or by the influence of solvents (lyotropic mesomorphism).

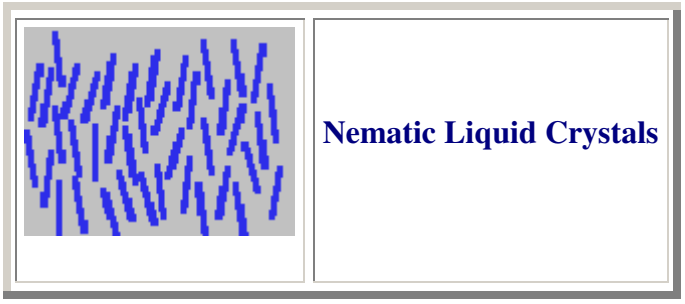


Figure 1.1: Cartoon of Nematic Phase of Liquid Crystals

Nematics Liquid Crystals are polarizable rod-like organic molecules on the order of 20 Angstroms in length. These molecules organize themselves in a parallel fashion, and demonstrate interesting and useful optical properties. Today, many more useful and interesting properties of nematics are known and exploited.

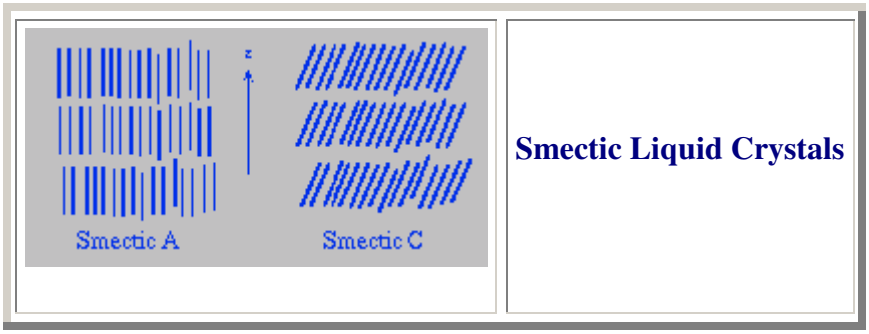


Figure 1.2: Cartoon of Smectic Phases of Liquid Crystals

Smectic Liquid crystals are different from nematics in that they have one more degree of orientational order than the nematics [3]. Smectics generally form layers within which there is a loss of positional order, while orientational order is still preserved. There are several different categories to describe smectics. The two best known of these are Smectic-A, in which the molecules align perpendicular to the layer planes, and Smectic-C, where the alignment of the molecules is at some arbitrary angle to the normal. There are also other possibilities, such as hexatic packing within layers, and termed hexatic-B, with a host of tilted variations.

There are many successful methods to study the Liquid Crystals at different phase transitions; some of them include calorimetry, x-ray, neutron, light scattering, NMR, optical microscopy, etc.

Among the different calorimetric techniques to study liquid crystals, traditional adiabatic calorimetry has high presicion and can be used to determine the latent heat at strongly first-order transitions. However, this method does not have enough resolution possibilities to characterize a second-order transition and requires a large sample of several grams to get reliable data [4].

Differential scanning calorimetry has high sensivity, making it a widely used method because of its ease of operation and small sample size (10 mg). Due to rapid scan rates of 1-5 K/min, the precision and absolute accuracy in enthalpy changes is modest, and the temperature resolution is poor [5,6].

Another calorimetric method able to achieve admirable temperature resolution and very good accuracy in measuring enthalpy changes in a proximity of a phase transition is adiabatic scanning calorimetry; although this method has a complex design and can operate using large samples (several grams) [7].

Since its introduction into the literature during the late 1960s [8], the AC calorimetry has developed into a very powerful, well-established and widely used technique for studying a variety of phase transitions in liquid crystals, polymers and biological systems. The power of the technique derives from the fact it involves a simple apparatus design; can operate very well with small samples (10-30 mg); very wide temperature range, from 50 mK to above 3000 K; a relative resolution of 0.2% or better in heat capacity is achieved [9]. The ac calorimetric technique is very sensitive to the internal and external relaxation times. Applications of this high-resolution technique include the effect of quenched random disorder introduced by the random fixed dispersion of solid surfaces, on phase structure and transitions [10,11,12]; low-temperature specific heat

measurements of superfluid helium films adsorbed in porous glasses, on graphite, and on superconducting films [13, 14, 15]; experimental evidence of continued melting of ethylene in graphite [16]; studies on phase transitions in bulk liquid crystals [17-21]; and in free-standing liquid crystal films [22]; studies on dynamic heat capacity of polymers, investigation of the melting kinetics in polycaprolactone [23]; simultaneous measurements of heat capacity and thermal conductivity of polymers, dynamic heat capacity and thermal conductivity of polymers measured in real time and in broad frequency region [24]. AC calorimetry is used successfully to investigate the thermal denaturation of diluted protein in solvent.

However, AC calorimetry has the limitation that it relies on conduction to apply heat to the sample (the sample is put in contact with the heater), and therefore can result in significant unwanted temperature gradients across the sample.

In this work we describe a modified and improved version of the ac calorimetry technique using Radio Frequency (RF)-Field-Heating. RF or dielectric heating is restricted to materials which are not good conductors, making it perfect for studying liquid crystals, polymers and biological systems. The sample to be heated is placed between the plates of a capacitor where it becomes the dielectric of the capacitor. A high frequency electromagnetic field causes the molecules of the dielectric to align millions of times per second, causing a heating mechanism similar to friction. The principal advantage of the RF-Field-Heating technique is that heat is generated evenly within the dielectric and not concentrated on the outside as with conduction heating technique. This eliminates the possibility of creating temperature gradients across the sample. We

present a general description of the physical experiment, then show typical data using this new technique in 8CB+aerosil sample to be consistent with previous investigations [10].

The study of ordering transitions in systems within disordered environments has provided new insights into the physics of phase transitions [25,26]. The isotropic (I) to nematic (N) and the nematic to smectic-A (SmA) phase transitions in the presence of the static (quenched) positional and orientational randomness provided by surface coupling at the silica-LC interfaces, can be investigated using liquid crystals (LC) incorporating low-density gels formed by silica aerosil dispersions (LC+A). Due to the first-order nature of the bulk transitions, where random disorder introduces a distribution of transition temperature shifts that broadens two-phase coexistence [27], as well as to the quadrupolar nature of the nematic order parameter, the I-N transition in LC+aerosil is a little difficult to interpret. A particular challenge is the observed doubling of the LC+aerosil calorimetric peak at the I-N transition in an intermediate range of silica concentrations [28-30]. This feature is interesting because it has no analogy in the LC bulk behavior, and thus its investigation provides a unique route to access the basic physics of discontinuous transitions in artificially disordered systems.

The three-dimensional nematic orientational order, established from the I state through a weakly first-order phase transition, is describable by a symmetric and traceless second rank tensor $\mathbf{Q}_{ij}$, which for uniaxial systems may be approximated on short length scales by a scalar order parameter (S) measuring the magnitude of order and on longer length scales by the director $\hat{n}$ describing the special variation of orientation axis [31]. The bulk transition exhibits significant pretransitional fluctuations and is very close to a tricritical point [32]. Both aspects are nontrivially affected by random disorder.

In this work we present an experimental study of LC+A samples with a low birefringence material, 4'-transbutyl-4-cyano-4-heptyl-bicyclohexane (CCN47), in which the biphenyls group is replaced by saturated hydrocarbon analogs, yielding a birefringence (Δn) about 1/10 that of cyanobiphenyls. Calorimetric and optical investigation of CCN47 LC+A samples through I-N transition over a range of silica densities are presented displaying the double I-N transition peak. This work shows compelling evidence that the I-N transition with weak quenched random disorder proceeds via a two-step process, in which random dilution is followed by random-field interactions on cooling from the isotropic phase, a previously unrecognized phenomenon.

This work is organized as follows. In Chapter 2, we give a general theoretical review of the liquid crystals phase transitions and their properties, and a review of the radio frequency or dielectric heating of liquid crystals. Chapter 3 is concentrated on the experimental techniques, AC calorimetry, non adiabatic scanning calorimetry, radio frequency heating calorimetry and sample materials. In Chapter 4 we present our results of radio frequency heating calorimetry technique in 8CB bulk and 8CB + aerosil samples at different silica densities. In Chapter 5 we present a high – resolution study of the isotropic to nematic phase transition of a low birefringence liquid – crystal compound incorporating an aerosil gel. A summary of the results and future work suggestions are presented in Chapter 6.

References:

[1] F. Reinitzer, Monatsch Chem., 9,421 (1888).

[2] O. Lehmann, Z. Physikal. Chem., 4, 462 (1889)

[3] Liquid crystals, Second Edition, S.Chandrasekhar, F.R.S. Centre for liquid crystals research, Bangalore.

[4] M.A.Anisimov, Critical Phenomena in Liquids and Liquid Crystals (Gordon and Breach, Philadelphial, 1991), Chap. 10.

[5] J.Thoen, Int. J. Mod. Phys. B 9, 2157 (1995).

[6] C.W.Garland, "Calorimetric Studies in Liquid Crystals: Physical Properties and Phase Transitions", edited by S. Kumar (unpublished).

[7] J.Thoen, H.Marijnissen, and W.Van Dael, Phys. Rev. A 26, 2886 (1982).

[8] P. F. Sullivan, G.Seidel, Phys. Rev. 173. 679. (1968).

[9] D. Finotello, S. Qian, G. S. Iannacchione, Thermochimica Acta 304/305 (1967) 303-316.

[10] G.S.Iannacchione, C.W.Garland, J.T.Mang, and T.P.Rieker, Phys. Rev. E 58, 5966 (1998).

[11] G.S.Iannacchione, S.Park, C.W.Garland, R.J.Birgeneau, and R.L.Leheny, Phys. Rev. E 67, 011709 (2003).

[12] A.Roshi, G.S.Iannacchione, P.S.Clegg, R.J.Birgeneau, Phys. Rev. E 59, 031703 (2004).

[13] Y.P.Feng, A.Jin, D.Finotello, K.A.Gillis and M.H.W.Chan, Phys.Rev., B38 (1998) 7041.

[14] H.B.Chae and M.Bretz, J.Low Temp.Phys., 76 (1989) 199.

[15] T.W.Kenny and P.L.Richards, Rev.Sci.Instrum., 61 (1990) 882.

[16] H.K.Kim, Q.M.Zhang and M.H.W.Chan, Phys.Rev.Lett., 56 (1986) 1579.

[17] J.E.Smaardyk and J.M.Mochel, Rev.Sci.Instrum., 49 (1978) 988.

[18] C.A.Schantz and D.L.Johnson, Phys.Rev., A17 (1978) 1504.

[19] D.L.Johnson, C.F.Hayes, R.J.DeHoff and C.A.Shantz, Phys.Rev., B18 (1978) 4902.

[20] J.D.LeGrange and J.M.Mochel, Phys.Rev., A23 (1981) 3215.

[21] J.Thoen, H.Marynissen and W.van Dael, Phys.Rev., A26 (1982) 2886.

[22] R.Geer, T.Stoebe, T.Pithford and C.C.Huang, Rev.Sci.Instrum., 62 (1991) 415.

[23] A. A. Minakov and C. Schick, Thermochimica Acta 330 (1999) 109-119.

[24] Minakov, A.A., Bugoslavsky, Yu. V., Schick, C., Thermochimica Acta, accepted (1998).

[25] M.Chan, N. Mulders, and J. Reppy, Phys. Today 49, No. 8, 30 (1996).

[26] T. Bellini, L. Radzihovsky, J. Toner, and N.A. Clark, Science 294, 1074 (2001).

[27] Y.Imry and M. Wortis, Phys. Rev. B 19, 3580 (1979).

[28] A. Roshi, Phys. Rev. E 69, 031703(2004).

[29] G. S. Iannacchione, Phys. Rev. E 58, 5966 (1998).

[30] P. Jamee, G. Pitsi and J. Thoen, Phys. Rev. E 66, 021707 (2002).

[31] P. G. de Gennes and J. Prost, The Physics of Liquid Crystals (Clarendon Press, Oxford, U.K., 1993), 2nd ed.

[32] A. Zywocinski, J. Phys. Chem. B 107, 9491 (2003).

CHAPTER 2

THEORY

2.1 Landau-DeGennes Theory of Phase Transitions

2.1.1 Nematic Phase

The Landau-DeGennes Theory of Phase Transitions is created based on some elegant and insightful assumptions in the early 1930s by Landau. This theory is applicable to the nematic, smectic, and the isotropic phases, and the transitions among them.

Molecules of liquid crystals may be seen as rigid rods whose long axis defines the orientation of the molecules, assuming these molecules are uniaxial, which is true for most of the thermotropic liquid crystals [1]. The nematic phase is characterized by an average orientation of the molecules along a particular direction $\hat{n}$, the director. The nematic phase possesses cylindrical orientation, because it is symmetric with respect to any rotation about the director $\hat{n}$.

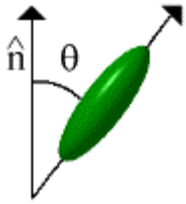


Figure 2.1: The angle between the director and the long axis of each molecule in the nematic phase

Quantification of the order in the system can be defined using an appropriate order parameter, a scalar Q, which measures the deviation from the director of a single molecule and is averaged over all the molecules in the system:

$$Q = \frac{1}{2}\left\langle 3\cos^2\theta - 1\right\rangle = \frac{3}{2}\left\langle \cos^2\theta\right\rangle - \frac{1}{2}, \qquad (2.1)$$

where θ is the angle between the long axis of the molecule and the director. The order parameter contains all the information necessary to describe the orientatioal order of most systems. If all the molecules are aligned in one direction, $\theta = 0$ for all the molecules, so $\left\langle \cos^2\theta\right\rangle = 1$ and the order parameter will have the value Q = 1. In an isotropic phase, molecules are randomly oriented in all possible directions, the term $\left\langle \cos^2\theta\right\rangle = \frac{1}{3}$, so the order parameter becomes Q = 0. If $\theta = \frac{\pi}{2}$, all the molecules are oriented perpendicular to the director, then $\left\langle \cos^2\theta\right\rangle = 0$ and so the order parameter becomes Q = -1/2.

The isotropic phase carries both translational and orientational symmetry and it is well known that in the nematic phase there exists a translational symmetry, but broken orientational symmetry. This indicates to us that the nematic to isotropic phase transition

should be first order, and thus, the first derivative of the free energy exhibits a discontinuity.

Writing the free energy density expansion in terms of the order parameter Q, and assuming the spatial variations of the order parameter [2], we can have:

$$f = f_0 + \frac{1}{2}AQ^2 - \frac{1}{3}BQ^3 + \frac{1}{4}CQ^4 + ... + D(\nabla Q)^2 + L(\hat{n}\cdot\nabla Q)^2 . \qquad (2.2)$$

where the order parameter **Q** is a uniaxial second rank tensor $\mathbf{Q}=\frac{1}{2}\mathrm{Q}(3n_i n_j - \delta_{ij})$ and an additional expansion in $\Delta\mathbf{Q} = \partial_k \mathrm{Q_{ij}}$ is added in the original expansion. The above expansion is a function of the spatial coordinates, and to find the total free energy of the system one has to integrate over the volume of the system. The parameter f_0 is not a function of the order parameter. The coefficients A, B, C, D, L are phenomenological parameters and, in general, all have temperature dependence. The total free energy is obtained by integrating f over the volume of the sample:

$$F(T) = \int_V f\left(T, \vec{r}\right) d^3 r . \qquad (2.3)$$

The state of the system can be found by finding the minimum of the free energy F with respect to the order parameter Q, using these conditions:

$$\frac{\partial F}{\partial Q} = 0 \qquad \text{and} \qquad \frac{\partial^2 F}{\partial Q^2} > 0 . \qquad (2.4)$$

In continuous phase transitions like SmA-N, the free energy given by Eq. (2.2) is unchanged under a Q = - Q transformation, thus no odd powers of Q will appear in the expansion for the free energy and the coefficient B will be 0. The transition becomes

first order as B changes from 0. Using the minimization conditions, B should become negative and the coefficient C should become positive.

Approaching a transition we can make the assumption that Q becomes very small (going to 0), and the free energy density is being ruled by the leading order term $\frac{1}{2}AQ^2$, considering the coefficients of the other terms constant. Very close to a transition, the free energy changes sign, so we can define the coefficient A as A = a(T-T*), where T* is the lowest temperature for isotropic stability (known also as the supercooling temperature), and the coefficient *a* is a phenomenological constant. Then, not counting spatial dependence of the order parameter, we can write the total free energy as:

$$F = Vf = V\left(f_0 + \frac{1}{2}a\left(T - T^*\right)Q^2 - \frac{1}{2}BQ^3 + \frac{1}{4}CQ^4 \right) \qquad (2.5)$$

The cubic term in Eq. (2.5) for the free energy makes the transition a first order for the mean field theory description. We can use this expression of the free energy to calculate many thermodynamic quantities. Using well known thermodynamic relations [3], the change in heat capacity of a sample is found to be:

$$\Delta C_p = \frac{aVT}{2}\left|\frac{\partial\left(Q^2\right)}{\partial T}\right| = aVTQ\left|\frac{\partial Q}{\partial T}\right|. \qquad (2.6)$$

Equation (2.6) permits a straightforward calculation of the heat capacity from the square of the order parameter Q.

2.1.2 Smectic Phase.

A natural choise for the order parameter of the N-SmA transition is a scalar equal to the amplitude fo the density modulation of the smectic phase. In the nematic phase Q

= 0, whereas in the smectic phase this becomes increasingly bigger. With this order parameter, and since the $\pm Q$ values differ only in the choice of the coordinative system, the free energy of the system can be written as:

$$f = f_0 + \frac{1}{2}aQ^2 + \frac{1}{4}bQ^4 + ... \qquad (2.7)$$

where, $a \cong \alpha(T - T_{NSmA})$. Below T_{NSmA}, a vanishes and above T_{NSmA}, a is positive. With this approximation alone one could have a second order transition.

However there are a lot of other complications that can not be neglected. A more thorough model must include in the free energy, all the effects that are considered separately in different models. The De Gennes model modifies the Landau simple model by adding the effects of the Frank elasticity energy and from the coupling to the amplitude of the nematic order parameter. With these additions the model exhibits first or continuous phase transitions and belongs to the 3D-XY Heisenberg magnet, universality class (the same with the normal to superconductor transition). So far no theory exists that can explain the richness of experimental findings. The critical exponents for the N-SmA transition are not quite the same as those of the 3D-XY universality class.

A very thorough compilation of the data available experimentally shows that the effective critical exponents, mostly lie between 3D-XY and tricritical values. Furthermore their behaviour is dependent on the McMillan ratio ($R_M = T_{NSmA}/T_{IN}$). In general the bigger McMillan ratio (the closer the T_{NsmA} is to T_{IN}) the more tricritical like the exponents become, in qualitative agreement with what McMillan suggested in his theory [13].

2.2 DISORDER EFFECTS.

2.2.1 QUENCHED RANDOM DISORDER MODELS.

In most cases, the random perturbations are introduced via the embedding of a random solid structure into the phase ordering material by the explicit confinement to a porous structure. The connection between the concentration and randomness of such solid inclusions and the strength of the random disordered field remains an open question. The physics of Quenched Random Disorder (QRD) is fundamentally understood by a random-field approach model [4]. The most well-known ones are the Ising and the Heisenberg models, which are models of magnetic spins in a lattice with nearest neighbor or longer order interactions. The Ising model is the simplest order-disorder model system with many theoretical applications. Onsager in 1944 solved the two-dimensional Ising model in zero magnetic field on a rectangular lattice. His work showed the first exact solution that exhibited a phase transition in a model with short-range interactions. No exact solution exists for the Ising model in three dimensions.

In the two-dimensional Ising model the points between pairs of spins of opposite signs can be connected to form boundary lines dividing the lattice into domains like Figure 2.2 shows.

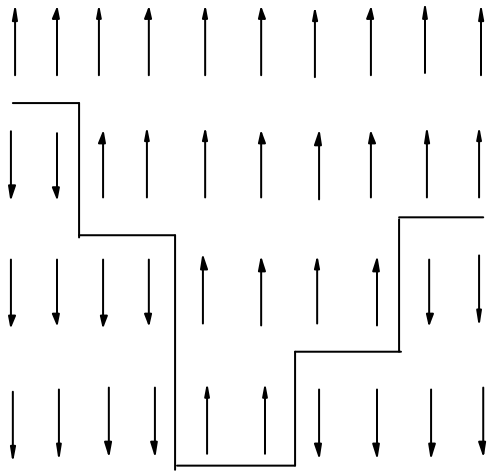


Figure 2.2: Example of a domain wall in the XY Ising model.

To find the net magnetization we have to consider the area of positive and negative domains, where the difference between these areas will be proportional to the net magnetization of the system. It is quite understandable that at T = 0 all the spins will point in one direction, there are no domain walls in the system. As T becomes positive, there will be enough energy in the system for the negative domains to appear, and the boundary lines will start to form. Still most of the spins will remain positive and the total magnetization of the system will be positive; the system is ferromagnetic. It is shown that magnetization becomes zero at a critical temperature $T_c > 0$.

The difference between the one and two dimensional models is that in one dimension, the existence and the length of each region can be changed without any cost of energy. The number of up and down spins on average will be the same. In two dimensions, at low temperatures the domains of down spins cannot grow, because they require longer boundaries, and thus more energy.

2.2.1.1 RANDOM-FIELD ISING MODEL

The Ising model is the simplest nontrivial model of magnetism. The spin at every site is either up or down and the interaction is between nearest neighbors only. The total energy can be expressed as:

$$H = J\sum_{\langle ij\rangle} S_i S_j - B\sum_i S_i \,, \tag{2.8}$$

where J represents the strength of exchange interaction between spins, B is a uniform external field applied to the system, S_i can have only two values ± 1, and i, j are the site indices. At low temperatures, the system is anti-ferromagnetic for J > 0 and ferromagnetic for J < 0.

For the Random-Field Ising Model (RFIM), the Hamiltonian of the system, Eq. (2.7), will include an additional contribution as:

$$H = J\sum_{\langle ij\rangle} S_i S_j - B\sum_i S_i - \sum_i h_i S_i \,, \tag{2.9}$$

where the last term represents the disorder effect, h_i, that fulfills the condition:

$$\langle h_i \rangle = 0, \ \langle h_i^2 \rangle = h_0^2 \tag{2.10}$$

For simplicity, the external uniform magnetic field is set to 0 hereafter. If the random field effect is too strong compared to the interactions between spins, $h_0 > |J|$, and as a result, every spin simply follows the respective site random field.

Let's consider now the weak disorder limit: $h_0 \le |J|$, where there is competition between the ordering effect of the interaction J and the disordered effect caused by random-field. Previous work done on this model shows that the presence of a random field impressively changes the nature of the transition. Onsager [5] showed that ordering

of the 2-dimensional Ising model is destroyed by the introduction of any finite random field.

2.2.1.2 RANDOM-FIELD XY MODEL

Random-Field XY Model (RFXYM) studies the role of quenched random disorder in systems with a planar XY symmetry, where the spins are allowed to rotate in a plane. In this case we must consider these spins as vectors, as opposed to the scalars in the Ising model. Many applications in type-II superconductivity, including high-Tc superconductors, have used this model as a starting point in understanding the vortex lattice structure. We can write the Hamiltonian of RFXYM starting from that of the Random Field-Ising Model, with the exception that the spins are considered as classical vectors:

$$H = J\sum_{\langle ij\rangle}\vec{S}_i \bullet \vec{S}_j - \sum_i B\vec{S}_i - \sum_i h_i\vec{S}_i \qquad (2.11)$$

Again we can consider the case of no applied uniform magnetic field for simplicity. Previous work [6] has shown that there is no long-range order in random-field XY magnets with quenched random disorder in less than four dimensions for arbitrary weak disorder. The nature of the ground state of the RFXYM is not yet well understood.

2.3 Review of Quenched Random Disorder effects in Liquid Crystals.

The study of the effect of quenched random disorder (QRD) on phase transition behavior remains an attractive area of research due to the broad implications outside the laboratory. The underlying physics has applications ranging from unique assemblies of complex fluids to doped semiconductors. Many systems have been the focus of both theoritical and experimental studies. The experimental efforts have concentrated on idealized model systems in the hopes of isolating the essential features of quenched random disorder. They include the still enigmatic superfluid transition of ^{4}He in aerogels and controlled porous glasses, the superfluid transition and phase separation of ^{4}He-^{3}He mictures in silica aerogels [14], and doped magnet systems [15]. Relatively recent efforts with liquid-crystals-silica compositions (using either aerogels or aerosils) [16, 17], have demonstrated that these are especially interesting model systems. Vycor-like controlled porous glasses have also been used to study liquid crystals [18, 19], but there are still some important questions as to the nature of the disorder. Liquid-crystals are of particular importance as a way to access “soft” (elastically weak) phases of continuous symmetry, which are directly coupled to surfaced and external fields.

First-order transitions with QRD have additional considerations than continuous transitions due to the possibility of two-phase coexistence (hence interfaces between ordered and disordered regions), intrinsically finite correlation length at the transition, and hysteresis effects. This has made the experimental and theoretical studies of quenched random disorder effects at first-order transitions much more challenging. Although the random-field model is the same starting point, first-order transitions have

the added complication of an energy penalty for the formation of interfaces between coexisting phases [20]. In this view, the QRD effect is as a random-field in each domain randomly shifting each domains transition temperature thus smearing the overall transition. However, nematics are very "soft" materials, and their elasticity can play an important role. A recent theoretical study, applied renormalization group analysis to the ordering of nematics with quenched random disorder concluded that such systems belong to the Random-Anisotropy (RA) Heisenberg class [21]. The observed suppression of the latent-heat of the first-order transition to nematic order with increasing QRD for 8CB+aerogel and other rigidly confined porous systems, appears to be consistent with the random transition temperature model [20], but the observed behavior for 8CB+aerosil is not and is perhaps more closely related to the RA-Heisenberg model [21].

In all fluid systems studied to date as models of QRD effects, including the liquid-crystal system, the random perturbations are introduced via the embedding of a random (gel-like) solid structure into the phase ordering material or by the explicit confinement to a porous structure. An open question remains as to the connection between the concentration and randomness of such solid inclusions and the strength of the random disordering field. Also, the identification of QRD is complicated by finite-size effects, which could, in principle, play a dominant role in such systems. In simple finite-size scaling (FSS), where the confining surfaces play no interactive role, the *bulk* critical correlation fluctuations are cut off at a length dictated by the distance between surfaces, which corresponds to a minimum reduced temperature where the transition is “truncated.” However, when the surfaces are arranged in a random manner with high void connectivity in order to introduce QRD, the distance between surfaces no longer acts

as an upper length scale in the system, and changes in the transition's critical behavior may also occur. Given the absence of LRO in such perturbed systems, the required characterization of the critical behavior may not be possible. In spite of this, if a critical power-law analysis of the transition heat capacity data is available, then, through two-scale universality, the critical behavior of the correlation length for $T > T^*$ may be estimated and compared with direct measurements. Finally, if the introduced random surfaces have in addition the freedom of an elastic response, then coupling between the gel and host elasticities can occur. This latter effect has only begun to be explored theoretically [22, 23].

2.4 THE DIELECTRIC PERMITTIVITY OF LIQUID CRYSTALS

In the various liquid crystal phases there exists long-range orientational order. The anisotropic elongated molecules are, on the average, aligned with their long axes parallel to each other. Macroscopically a unique axis is defined in this way. In the nematic phase the molecules translate freely as in the isotropic liquid; the centers of mass are distributed at random. In the smectic phases there is additional positional ordering in layers. In the Smectic-A phase the layers are perpendicular to the preferred direction. Within layers the distribution of the centers of mass is again random.

Because of the uniaxial symmetry in the nematic and smectic-A liquid crystals, the dielectric permittivity differs in value along the preferred axis ($\varepsilon_{\parallel}$) and perpendicular to the axis ($\varepsilon_{\perp}$). The dielectric anisotropy is defined as $\Delta\varepsilon = \varepsilon_{\parallel} - \varepsilon_{\perp}$. In order to measure

$\varepsilon_{\parallel}$ and $\varepsilon_{\perp}$ the preferred direction has to be uniform over the whole sample. Usually a plane capacitor is used with a nematic layer of about 100 μm as dielectric. With suitable surface treatments the preferred axis can be made to lie either parallel or perpendicular to the electrodes. A sample with a uniform preferred axis can also be obtained by applying a magnetic field. This has the advantage that $\varepsilon_{\parallel}$ and $\varepsilon_{\perp}$ can be measured on the same sample, which increases the accuracy to which $\Delta\varepsilon$ is obtained. The preferred axis in smectic liquid crystals is rather difficult to influence. This is the main reason why dielectric data on smectic are scarce.

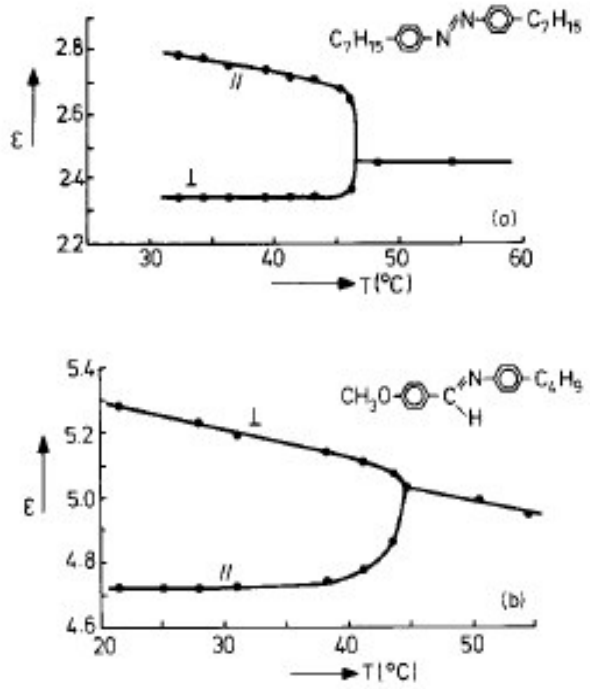


Figure 2.3: Permittivities in nematic and isotropic phase of a typical nonpolar compound (a) and a typical polar compound (b).

The sign and magnitude of $\Delta\varepsilon$ are of utmost importance for the applicability of the various electrooptic effects. Figure 2.3 gives some typical results of dielectric

measurements. For nonpolar molecules $\Delta\varepsilon$ is of order of 0.4. In case of N-(p-methoxybenzylidene)-p-n-butylaniline (MBBA) a negative anisotropy is found. In practice anisotropies between, say, +1 and –1 are often observed. Compounds with a large anisotropy can be synthesized by substitution of a strongly polar group (for example a cyanide group) in specific positions. Examples of the largest positive $\Delta\varepsilon = +30$ and $\Delta\varepsilon = -5$, the larges negative anisotropy observed so far can be found.

The dielectric anisotropy $\Delta\varepsilon$ is proportional to the order parameter in the liquid crystal system: $\Delta\varepsilon = \varepsilon_{\|} - \varepsilon_{\perp} \sim Q$. The order parameter as well as the dielectric anisotropy decrease rapidly with rise of temperature.

The dielectric constants are, of course, frequency dependent. The dipole orientation part of the polarization parallel to the preferred direction (*I*-direction) may be expected to be characterized by a relatively long relaxation time. This arises because of the strong hindering of the rotation of the longitudinal component of the dipole moment about a transverse axis. On the other hand the orientational polarization along the 2-direction will have a much faster relaxation, comparable to the Debye relaxation in normal liquids, as this involves rotation about the long axis of the molecule. If there are additional dipoles in parts of the molecule, with their own internal degrees of rotation, the corresponding relaxation times will again be similar to that in a liquid. The expected form of the dispersion curves for a compound like p-azoxyanisole is illustrated in Figure 2.4.

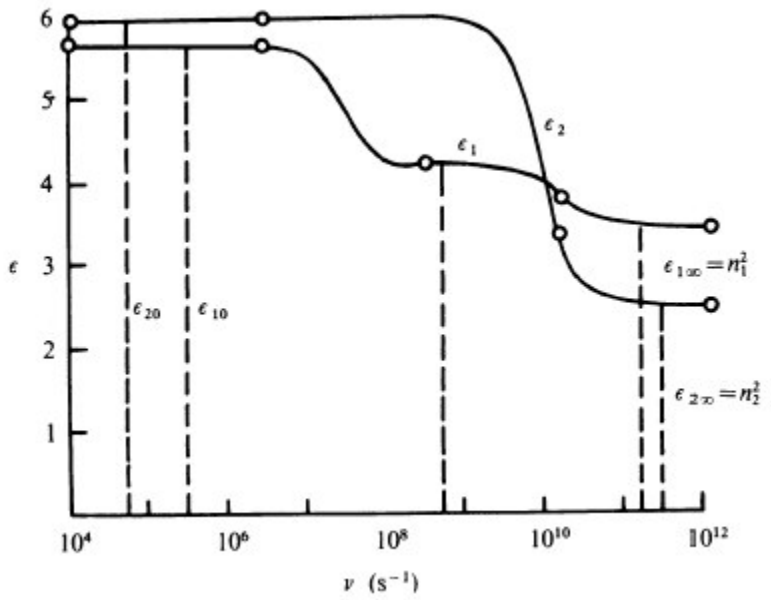


Figure 2.4: Expected form of the dispersion of the principal dielectric constants of 4,4'-di-n-alkoxyazobenzenes. The suffix 0 refers to the static values and the suffix ∞ to the optical values. ε_1 shows the low frequency relaxation and both ε_1 and ε_2 show the normal Debye high frequency relaxation.

2.5 RADIO FREQUENCY HEATING OF DIELECTRICS.

There are three different kinds of heating mechanisms of the materials: induction heating, dielectric heating and conduction heating. In the induction heating, the material to be heated is an electric conductor and placed in an induction coil. A high frequency electromagnetic field is induced by passing a high frequency electric current through the coil. The magnetic field will induce eddy currents in the material, and the resistance of

the material to the flow of eddy currents will make the material to heat up. The depth at which the current flows depends on the frequency of the magnetic field; higher frequencies will 'produce' shallow depths, lower frequencies will 'produce' deeper depths.

Conduction heating is the most well-known method of heating. In this method the material to be heated is put in contact with a heat source and a heat flow from the heat source to the material will occur. Thermal conduction is associated also with the energy transfer process between 2 materials that are at different temperatures. The heating process can be seen on an atomic scale as an exchange of kinetic energy between microscopic particles-molecules, atoms, electrons- in which less energetic particles gain energy in collisions with more energetic particles. Metals are typically very good conductors, because there are a large number of free electrons which move throughout the metal, so they can transport energy over quite large distances. In contrary, gases are poor conductors because the distances between molecules are relatively big.

Radio Frequency Heating refers to the use of electromagnetic waves of certain frequencies to generate heat in a material. Radio frequency electromagnetic waves usually cover the frequency spectrum from 30 to 300 MHz, and can be absorbed and converted to heat in nonmetallic materials known as "lossy" dielectrics, similar to microwave heating. This is the reason why both radio frequency and microwave heating are known as dielectric heating. For RF heating, the sample to be heated is placed between the plates of a capacitor where it becomes the dielectric of the capacitor. A high frequency electromagnetic field is applied across the plates of the capacitor which will cause the material inside to be heated through two mechanisms. The first and most

dominate mechanism is due to the high frequency electric field, which will force the polar molecules already present in the material to align millions of times per second causing heating through friction. The second heating mechanism is the induction of the new dipole molecules inside the material as the result of the applied electric field across the plates of the capacitor. The molecular polarization can lead to heating via distortion of the dielectric molecules.

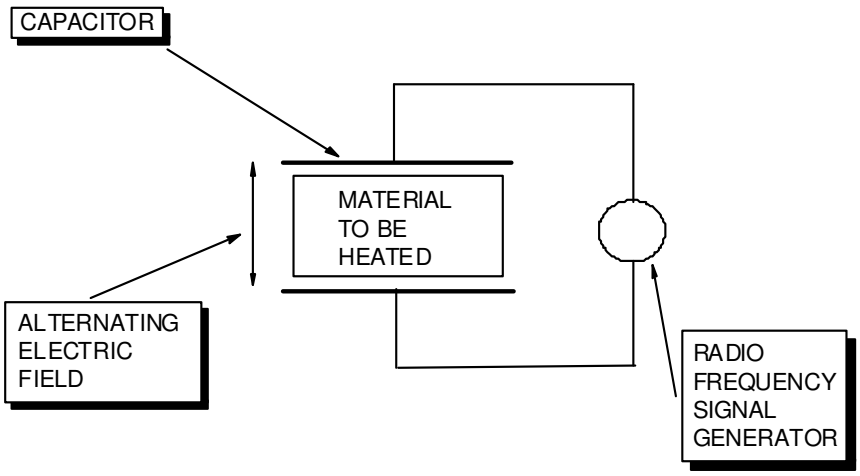


Figure 2.5: Simple Radio Frequency applicator.

The advantages of radio frequency heating include:

Rapid Heating:

In Radio Frequency Heating, the power dissipated inside the material can vary from 1 to 20 W/cm^3, which will cause a very high heating rate. The heat is generated evenly within the material and not concentrated on the outside as with conventional

heating methods, so, as a result, the high heating power applied to the material will not destroy it.

Uniform Heating:

In Radio Frequency heating, the heating of the material occurs immediately and uniformly throughout the material to be heated, in contrary with the other conventional methods where heating relies on conduction to transfer the heat from the surface to the center of the material. This will result in unwanted temperature gradients across the sample.

Selective heating:

This is a very important feature of the Radio Frequency Heating. If we are dealing with samples that have a mixture of different materials, the heating power will concentrate on the material which has the highest dielectric loss factor. This is called selective heating, and can be used in many applications. This selective heating is used very efficiently in industry; for example in controlling the moisture levels in the paper and board applications, where the radio frequency heating concentrates itself on the wet spots and in a very short amount of time controls the moisture levels in these materials.

Instant Control:

By regulating the power applied, radio frequency heating can be instantly controlled and accurately regulated, so it can result in the safe and precise control of the applied heat.

Many industrial processes employ radio frequency heating in their applications. Below are listed some of the many applications using radio frequency heating as a very powerful and accurate method of heating different materials: textile drying, paper making

and converting, baking operations, chipboard preheating, dye fixation, pasteurization/sterilization of chilled foods, tobacco, plastic and polymers.

Materials that can be heated using radio frequency heating:

The ability of radio frequency to heat a dielectric material will depend on the capability to dissipate electromagnetic energy into thermal energy. This capability will depend on many factors, like chemical-physical characteristics of the material, its temperature, and the strength of the applied electromagnetic field. The efficiency of heating the material using radio frequency heating is directly connected to a value known as “the loss factor” of the material. The higher this loss factor, the more electromagnetic energy is absorbed and transformed into heat. Below is a list of loss factors of some well-known materials at 298 K, and under 1 MHz electromagnetic field frequency.

Carbon tetrachloride (CCl4)	0.00009
Teflon	0.0004
Polythene	0.008
PVC (pure)	0.06
PVC (plasticised)	0.4
Wood	0.01-0.08
Paper	0.1
Ureic, Phenolic and Melamminic resins	0.2
Methanol	6.2
Fats	24
Water	100

Table 2.1: The loss factor of some materials (298 K, 1 MHz).

To calculate the heating produced when a dielectric material is inserted between the plates of a capacitor, which is subject to an alternating electric field, we should consider the electrical phenomena and electric factors involved. Let's recall some of the relations between electrical parameters in AC circuits.

In a pure capacitance circuit, the current leads the applied voltage by 90^0. Because of the loss in the capacitance the current will lead the voltage by less than 90^0. Below are shown these relations between current and voltage in these cases:

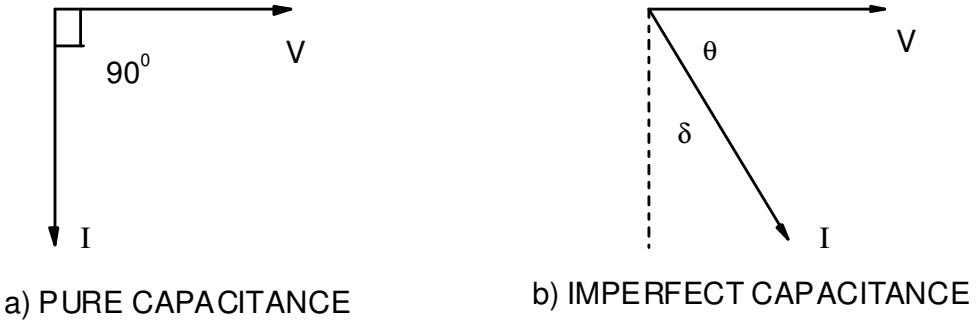


Figure 2.6: Relations between the current and voltage in a pure capacitance circuit a) and in an imperfect capacitance circuit b).

We will be considering the imperfect capacitance circuit. In the imperfect capacitance circuit, the angle δ is known as the loss angle, and $\cos\theta$ is the power factor. We can resolve the vector of the current I into two components: one in phase with the applied voltage, component I_R, and one 90^0 out of phase, component I_C, leading to the equivalent circuit and current-voltage relationships as below:

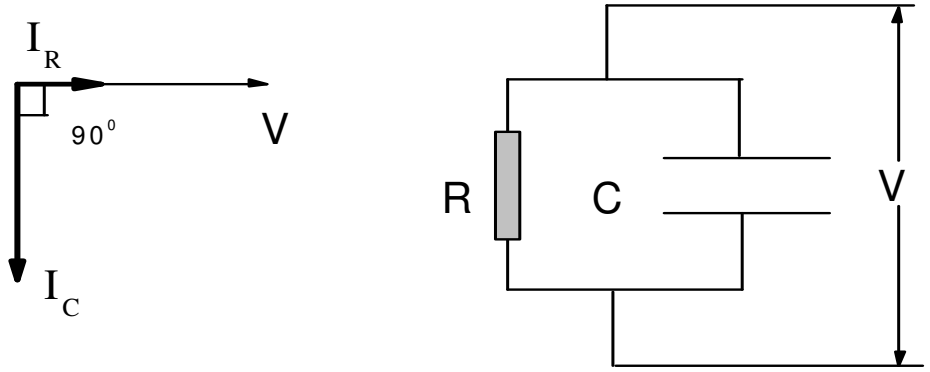


Figure 2.7: The equivalent circuit and the current-voltage relationships for an imperfect capacitance circuit.

where R is the equivalent resistance of the imperfect capacitance, C is the value of the capacitance, I_R is the current through R, and I_C is the current through C.

The following relations are true for I_R and I_C:

$$I_R = I\sin\delta \qquad \text{and} \qquad I_C = I\cos\delta\,. \tag{2.12}$$

The power loss can occur only in resistive components where the current and voltage are in phase, thus the total power will be given by:

$$P = \left(I_R^{\,2}\right)R - \frac{V^2}{R}\,. \tag{2.13}$$

If we denote X_C as the reactance of the capacitor C, then the following relations are also true:

$$I_C = \frac{V}{X_C}, \qquad X_C = \frac{1}{2\pi fC}, \qquad \text{and} \quad I_C = 2\pi fCV\,. \tag{2.14}$$

Also, we can calculate the ratio of I_R/I_C, which is a very important factor, **the loss factor** of the material inside the capacitor:

$$\tan\delta = \frac{I_R}{I_C} = \frac{\frac{V}{R}}{2\pi fCV} = \frac{1}{2\pi fCR} \tag{2.15}$$

From Eq. (2.14) we can express R in terms of *f*, C and the loss factor $\tan\delta$:

$$R = \frac{1}{2\pi fC(\tan\delta)} \tag{2.16}$$

Then the power absorbed by the dielectric inside the plates of the capacitor will be given by:

$$P = \frac{V^2}{R} = 2\pi fC(\tan\delta)V^2. \tag{2.17}$$

If we express the capacitance C in terms of the dielectric constant of the dielectric between the plates of the capacitor:

$$C = \frac{A|\varepsilon|}{d}, \tag{2.18}$$

where $|\varepsilon|$ is the permittivity of the dielectric inside the plates of the capacitor and can be expressed as:

$$|\varepsilon| = K\varepsilon_0, \tag{2.19}$$

where ε_0 is the permittivity of the vacuum and has the value $8.8542\times10^{-12}\frac{C^2}{Nm^2}$, and K is called the dielectric constant of the material, A is the area of the capacitor plates, *d* is the distance between the plates.

Inserting Eq. (2.17) into (2.16) gives:

$$P = 2\pi f\frac{A}{d}(\tan\delta)|\varepsilon|V^2 = \frac{A}{d}\omega_d(\tan\delta)|\varepsilon|V^2, \tag{2.20}$$

where ω_d is the angular frequency of the applied alternating electric field. Equation 2.19 represents the power absorbed by a dielectric material placed inside a capacitor driven by a voltage V between its plates.

One of the most familiar and important electric systems is the one made up of a capacitor C, an inductor L, and a resistor R in series. Such a system, called a damped driven oscillator can produce oscillations involving the periodic transfer of energy between the capacitor and the inductor, with a continual dissipation of energy in the resistor.

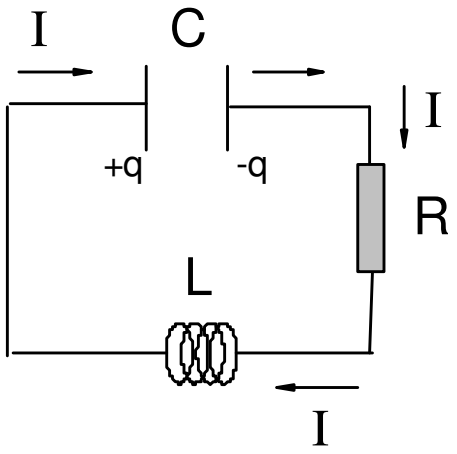


Figure 2.8: An LCR series circuit.

It can be easily shown [7] that the voltage across the plates of the capacitor for a driven damped oscillator can be expressed as:

$$V(\omega_d) = \frac{V_0}{\sqrt{\left(1-\left(\frac{\omega_d}{\omega_0}\right)^2\right)^2 + \left(\frac{\omega_d}{Q\omega_0}\right)^2}}, \qquad (2.21)$$

where ω_d is the driving angular frequency of the LRC series circuit, V_0 is the driving voltage, ω_0 is the natural frequency of oscillations of the LRC series circuit, and Q is the quality factor of the circuit. The larger the value of Q, the less the dissipative effect and the greater the number of cycles of free oscillation for a given decrease of amplitude.

Substituting Eq. (2.20) into (2.19) obtains the expression for the power absorbed by the dielectric inside the capacitor driven by a voltage V_0, at a driven frequency ω_d:

$$P = \frac{A}{d}\omega_d (\tan\delta)|\varepsilon| \frac{V_0^{\,2}}{\left(1-\left(\frac{\omega_d}{\omega_0}\right)^2\right)^2 + \left(\frac{\omega_d}{Q\omega_0}\right)^2}. \tag{2.22}$$

As apparent from Eq. (2.21), the power absorbed depends on many factors, such as the driving frequency, the driving voltage, the loss factor, the capacitance, and the quality factor of the circuit. In the resonance condition $\omega_d = \omega_0$, the power being delivered will be maximum given by:

$$P(\omega_0) = \frac{A}{d}\omega_0 (\tan\delta)|\varepsilon| Q^2 V_0^{\,2}, \tag{2.23}$$

where ω_0 is the natural frequency of oscillations of the circuit given by:

$$\omega_0 = \frac{1}{\sqrt{LC}}. \tag{2.24}$$

Building a circuit with a resonance frequency convenient and driving the voltage at the resonance frequency, will yield the maximum power delivered to the material placed between the plates of the capacitor.

An important factor appearing in the expression for the power absorbed is the permittivity of the dielectric material. Debye [8] has shown a relationship between

dielectric constant, hence the permittivity, and the loss factor and frequency. The complex permittivity is given by a complex quantity:

$$\varepsilon = \varepsilon_1 + i\varepsilon_2 , \tag{2.25}$$

where ε_1 is the real and ε_2 is the imaginary part of the permittivity. It is shown that:

$$\varepsilon_1 - \varepsilon_\infty = \frac{\varepsilon_0 - \varepsilon_\infty}{1+\omega^2\tau^2} \quad \text{and} \quad \varepsilon_2 = \frac{(\varepsilon_0 - \varepsilon_\infty)\omega\tau}{1+\omega^2\tau^2}, \tag{2.26}$$

where: ω is the angular frequency, τ is the relaxation time, ε_0 is the static permittivity, and ε_∞ is the permittivity at very high frequencies.

Debye has derived the formula for the relaxation time considering a dielectric material of spherical molecules moving in a viscous medium:

$$\tau = \frac{4\pi\eta a^3}{kT}, \tag{2.27}$$

where: η is the viscosity of the medium, *a* is the radius of the spherical molecules, *k* is the Boltzman constant, and T is the absolute temperature.

Also, the loss factor can be expressed in terms of the imaginary and real parts of the permittivity:

$$\tan\delta = \frac{\varepsilon_2}{\varepsilon_1} = \frac{\varepsilon_0 - \varepsilon_\infty}{\varepsilon_0 + \varepsilon_\infty}. \tag{2.28}.$$

We can substitute the Eq. (2.27) for the loss factor into Eq. (2.22) and obtain an expression for the power absorbed by the dielectric inside the capacitor driven by a voltage V_0, at a driven frequency ω_d:

$$P(\omega_0) = \frac{A}{d}\omega_0 \frac{\varepsilon_2}{\varepsilon_1}|\varepsilon| Q^2 V_0^{\,2}. \tag{2.29}$$

References:

[1] M. A. Anisimov, Critical Phenomena in Liquids and Liquid Crystals, (Gordon and Breach, 1991).

[2] P. Sheng and E. B. Priestley, Chapt. 10, Introduction to Liquid Crystals, edited by E. B. Prestley, P. J. Wojtowicz and P. Sheng, (Plenum Press, 1974).

[3] M. W. Zemansky, Heat and Thermodynamics, 6th edition (McGraw-Hill, 1981).

[4] A. Pelissetto and E. Vicari, Phys. Rev. B 62, 6393 (2000).

[5] L. Onsager, Phys. Rev. 65, 117 (1944).

[6] Y.Imry and S. K. Ma, Phys. Rev. Lett. 35, 1399 (1975).

[7] A. P. French, Vibration and Waves, The M. I. T. Introductory Physics Series, W. W. Norton & Company, New York – London (1971).

[8] B. Debye, "Polar Molecules", The Chemical Catalog Co., (1929).

[9] Chandrasekhar, S., Liquid Crystals, 2nd edition (1992)

[10] Robert Brout, Phase Transitions, W. A. Benjamin, Inc., New York, Amsterdam (1965)

[11] C. Domb, M. S. Green, Phase Transitions and Critical Phenomena, Vol. 1, Academic Press, London New York (1972)

[12] P. G. de Gennes, The Physics of liquid crystals, Clarendon Press, Oxford, London (1979)

[13] W. L. McMillan Phys. Rev. A., vol 7, p. 1419, 1973.

[14] M. Chan, N. Mulders, and J. Reppy, Physics Today 49, 30 (1996).

[15] Q. J. Harris, Q. Ferris, Y. S. Lee, R. J. Birgeneau, and A. Ito, Phys. Rev. Lett. 78, 346 (1997).

[16] G. S. Iannacchione, C. W. Garland, J. T. Mand, and T. P. Rieker, Phys. Rev. E 58, 5966 (1998).

[17] T. Bellini, L. Radzihovsky, J. Toner, and N. A. Clark, Science 294, 1074 (2001).

[18] Z. Kutnjak, S. Kralj, G. Lahajnar, and S. Zumer, Fluid Phase Equilibra (2003).

[19] G. S. Iannacchione, G. P. Crawford, S. Qian, J. W. Doane, D. Finotello, and S. Zumer, Phys. Rev. E 53, 2402 (1996).

[20] Y. Imry and M. Wortis, Phys. Rev. B 19, 3580 (1979).

[21] D. E. Feldman, Phys. Rev. Lett. 84, 4886 (2000).

[22] L. Radzihovsky, private communications.

[23] P. D. Olmsted and E. M. Terentjev, Phys. Rev. E 53, 2444 (1996).

CHAPTER 3

EXPERIMENTAL TECHNIQUES

3.1 REVIEW OF AC CALORIMETRY.

The heat capacity of a substance is defined as the amount of energy needed to raise the temperature of that substance by 1 K. The heat capacity is a measure of how thermally stable a substance is to the addition of energy and is a suitable quantity to determine the behavior of phase transitions. AC calorimetry technique is a very powerful, well-established and widely used technique for studying a variety of phase transitions in condensed and soft-condensed matter systems. The AC calorimetry technique has many advantages over other techniques that measure the heat capacity, because of its characteristics:

- the measurements of heat capacity are taken under near equilibrium conditions, which is a very important fact because all the thermodynamic theory of phase transitions is based upon equilibrium conditions.

- Total computer automation leads to very high relative resolution of better than 0.06% in measuring heat capacity.
- It can achieve very high relative caloric sensitivity, using very small amounts of sample 10-20 ml. This is a very important factor for two reasons: one, to ensure thermal equilibrium, and secondly, sometimes large quantities of samples may not be accessible.
- Thermal isolation of the sample from the surroundings (like in the adiabatic methods) is not required.

3.1.1 Theory of Operation.

The basics of the AC calorimetry technique consist of applying periodically modulated, sinusoidal power, and measuring the resulting sinusoidal temperature response. As we will show later, the heat capacity of the sample is inversely proportional to the amplitude of the temperature oscillations. The derivation of the basic operating equations based on a simplified thermal model of the system, is meeting certain requirements geometry independent. Geometry independence means that the location of the heater, and the location of the thermometer measuring the temperature of the sample, are not important. A simple thermal model used in AC calorimetry technique, based on the model worked out by Sullivan and Seidel [1], called the one-lump-thermal model, consists of number of elements: a cell heater with heat capacity C_h, which has a thermal conductance K_h to the cell, a thermometer with heat capacity C_θ, attached to the cell through a thermal conductance K_θ, a sample with heat capacity C_S linked to the cell by

thermal conductance K_S (ususally, this is the sample's internal thermal conductance), and the cell itself which holds the sample with heat capacity C_C. All these elements are linked to a controlled bath at temperature T_b through a thermal conductance K_b.

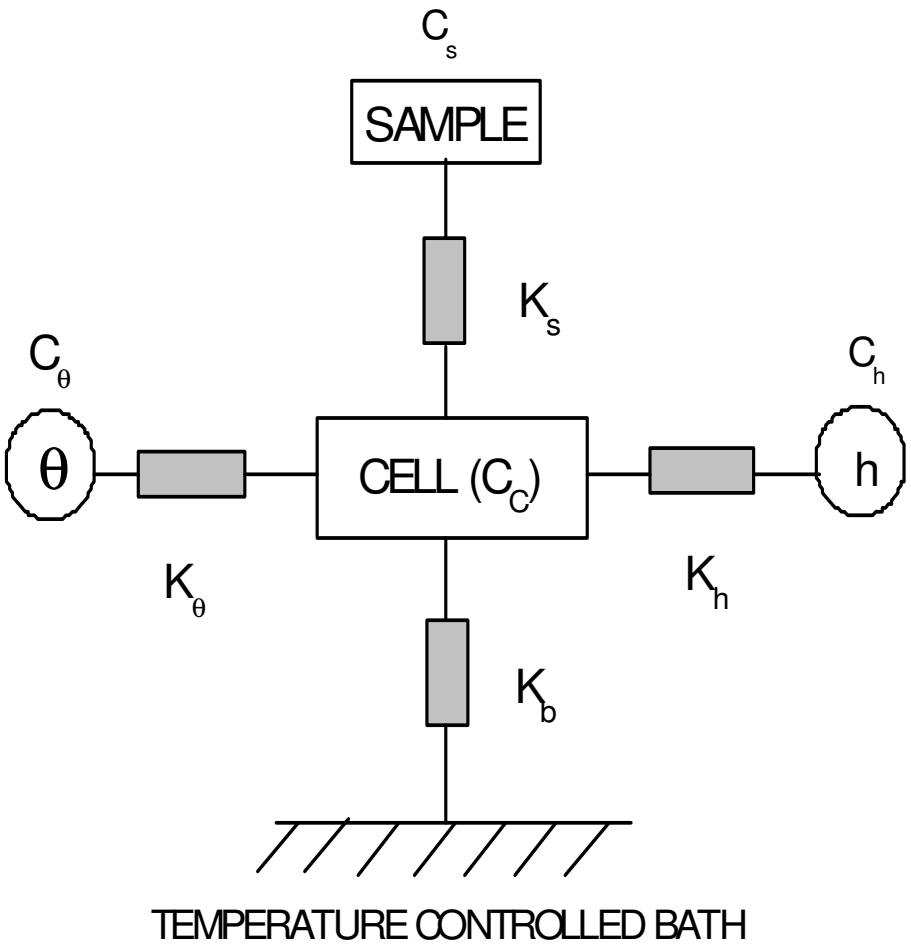


Figure 3.1: One lump thermal model used in an AC calorimetry.

The total heat capacity of the system cell+sample+heater+thermistor is given by:

$$C = C_h + C_\theta + C_S + C_C \,. \tag{3.1}$$

The thermal relaxation time constants respectively for the thermistor, heater and sample, are defined as :

$$\tau_\theta = \frac{C_\theta}{K_\theta} \qquad \tau_h = \frac{C_h}{K_h} \qquad \tau_S = \frac{C_S}{K_S} \,. \tag{3.2}$$

The external and internal thermal relaxation times for the system are defined as:

$$\tau_e = \frac{C}{K_b} \text{ , external;} \qquad \tau_i^{\,2} = \tau_\theta^{\,2} + \tau_h^{\,2} + \tau_S^{\,2} \text{ , internal.} \tag{3.3}$$

The external time constant represents the time for the cell+sample+heater+thermometer system to reach thermal equilibrium with the bath, and the internal time constant represents the time for the entire assembly to reach equilibrium with the applied heat.

The heating power applied to the sample, Joule's heat, is generated by sending an oscillatory sinusoidal voltage through a resistive wire, and can be expressed as:

$$\dot{Q} = \dot{Q}_0 \cos^2(\omega_v t) = \frac{1}{2}\dot{Q}_0\left(1 + \cos(2\omega_v t)\right), \tag{3.4}$$

where $\dot{Q}_0$ is the heating amplitude, and $\omega_V = 2\pi f_v$ is the angular heating voltage frequency. Temperature oscillations will be at twice the voltage frequency: $\omega = 2\omega_V$.

Starting from the very simple energy balance equations:

Input Heat = Heat absorbed - Heat leaked,

for every element in the setup, we can write the following heat balance equations for the heater, sample and thermometer [2]:

$$\begin{cases} C_h \dfrac{\partial T_h}{\partial t} = \dfrac{\dot{Q}_0}{2} + \dfrac{\dot{Q}_0}{2}\cos(\omega t) - K_h (T_h - T_S) \\ C_S \dfrac{\partial T_S}{\partial t} = K_h (T_h - T_S) - K_b (T_S - T_b) - K_\theta (T_S - T_\theta) \\ C_\theta \dfrac{\partial T_\theta}{\partial t} = K_\theta (T_S - T_\theta) \end{cases} . \qquad (3.5)$$

Solving these equations, we can find the following expression for the temperature of the thermometer:

$$T_\theta = T_b + \frac{\dot{Q}_0}{2K_b} + \frac{\dot{Q}_0}{2\omega C}\left(1 + \frac{1}{(\omega\tau_e)^2} + \omega^2\left(\tau_\theta{}^2 + \tau_h{}^2 + \tau_S{}^2\right)\right)^{-\frac{1}{2}} \cos(\omega t - \alpha). \qquad (3.6)$$

The first term in Eq. (3.6) is the bath temperature, the second term is the temperature T_{dc}, coming from the *rms* heating, and the third term is the induced temperature oscillation. As we can see the amplitude of these oscillations is given by:

$$T_{ac} = \frac{\dot{Q}_0}{2\omega C}\left(1 + \frac{1}{(\omega\tau_e)^2} + \omega^2\left(\tau_\theta{}^2 + \tau_h{}^2 + \tau_S{}^2\right)\right)^{-\frac{1}{2}}, \qquad (3.7)$$

and the phase shift α between the applied heat and the resulting temperature oscillations will be given by:

$$\alpha = -\frac{\pi}{2} + \arctan\left(\frac{1}{\omega\tau_e} - \omega\tau_i^*\right), \qquad (3.8)$$

where τ_i^* is a time constant defined as: $\tau_i^* = \tau_h + \tau_S + \tau_\theta$.

Eq. (3.7) is the exact expression for the temperature oscillations. If the following conditions apply:

$$\frac{1}{\tau_e} < \omega < \frac{1}{\tau_i}, \tag{3.8}$$

then we can write the simplified solution for Tac:

$$T_{ac} \cong \frac{\dot{Q}_0}{2\omega C}, \tag{3.9}$$

and the simplified solution for the total heat capacity will be given by:

$$C \cong C^* = \frac{\dot{Q}_0}{2\omega T_{ac}}. \tag{3.10}$$

We can use the Eq. (3.10) for the total heat capacity, if the conditions in Eq. (3.8) are satisfied. We can fulfill Eq. (3.8) by adjusting external and internal time constants. The thermal link from the sample to the bath can be used to control the external time constant, by choosing the appropriate material, length and cross-section area of the electrical leads for the heater and thermometer. For the internal time constant the most important role is played by the sample thickness, since the time constants of the thermometer and heater are generally very short and temperature independent. The requirements in Eq. (3.8) can be fulfilled if the sample thickness is less than the thermal diffusion length, which can be expressed as:

$$l = \sqrt{\frac{2K_S A}{\omega C_S}}, \tag{3.11}$$

where A is the cross-sectional area of the sample. If these conditions are filled, then the system is geometry independent, and the locations of the heater and the thermometer become unimportant. If the thickness of the sample is greater than the thermal diffusion length, then the sample will not reach equilibrium with the applied heat, and will not follow the imposed temperature oscillations.

We can verify if Eq. (3.8) are satisfied experimentally, by performing a frequency scan. The frequency scan can be done by varying the frequency of the applied heat, keeping all the other parameters constant, like the power, the bath temperature, and recording the amplitude of the temperature oscillations. A log-log plot of ωT_{ac} vs. ω, will show a frequency independent "plateau." This is the region where T_{ac} is much greater than the internal time constant, and much smaller than the external time constant, so the requirements 3.8 are satisfied. A typical frequency scan taken with the calorimeters operating in the lab are shown in Figures 3.2 and 3.3.

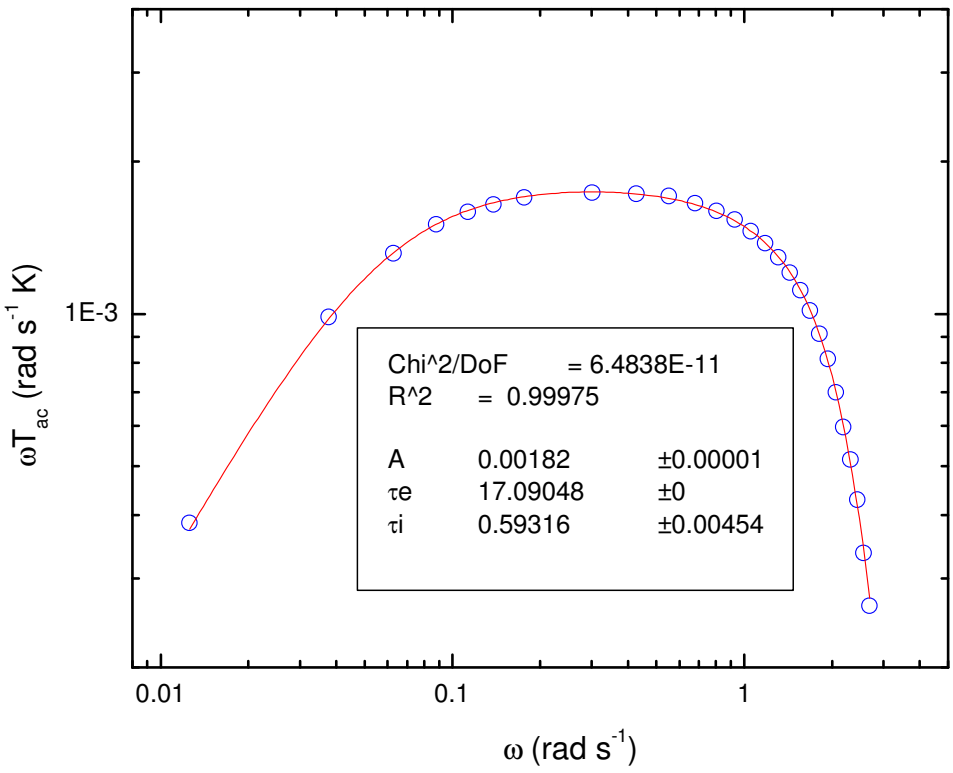


Figure 3.2. Log-log plot of ωTac vs. ω. The solid line shows a fit of the data. The scan was performed at T = 310 K.

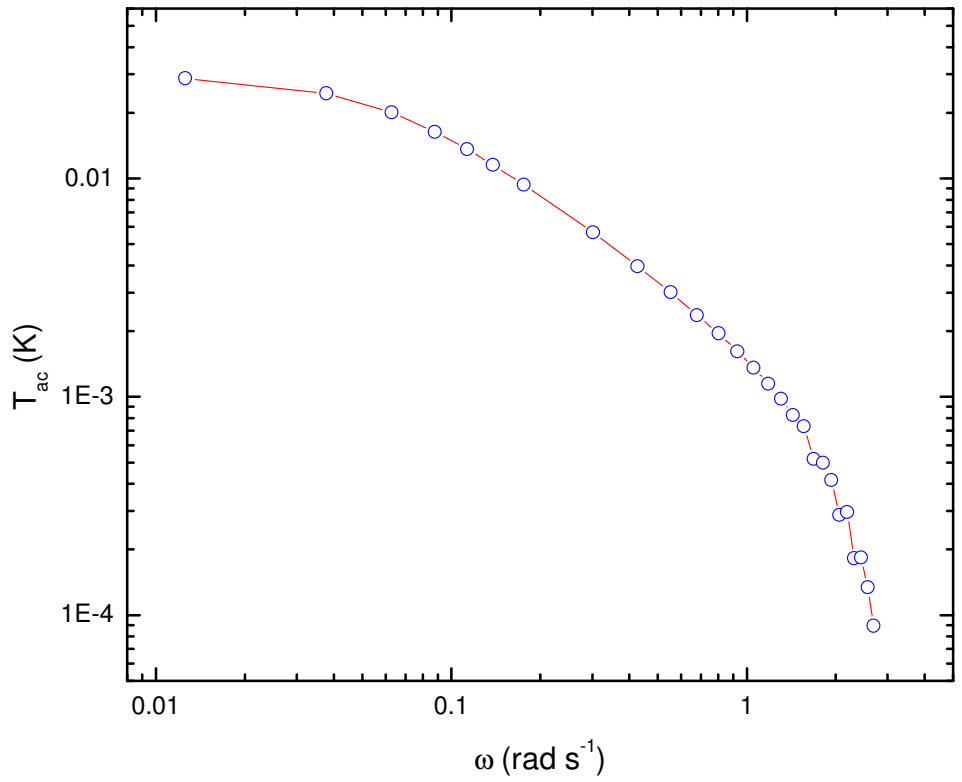


Figure 3.3. Log-log plot of Tac vs. ω.

The Figure 3.2 shows also the results from the fitting of ωT_{ac} vs. ω data, using the following fitting equation:

$$\omega T_{ac} = \frac{A}{\sqrt{\left(1+\left(\omega\tau_e\right)^{-2}+\left(\omega\tau_i\right)^{2}\right)}}. \qquad (3.12)$$

The resulting parameters are:

A = 0.00182, τ_e = 17.09 sec., τ_i = 0.593 sec.

Figure 3.3 shows the log-log plot of T_{ac} vs. ω data. As ω goes to zero, T_{ac} will saturate to the T_{dc} value of the temperature response, which can be used to calibrate the power to the cell.

From Eq. (3.6), we also see that there is a phase shift between the applied heat and the resulting temperature oscillations, given by Eq. (3.8). Usually the use of the reduced phase shift is preferred, which is constructed by removing the $-\frac{\pi}{2}$ term, and is given by:

$$\phi = \alpha + \frac{\pi}{2} = \arctan\left(\frac{1}{\omega\tau_e} - \omega\tau_i^*\right). \qquad (3.13)$$

From previous work [3-5], it is shown there is a typical phase shift signature associated with the order of the phase transition. At first-order phase transitions, latent heat is released in the system, and there is an additional phase shift showing up, which is the reason for the existence of the narrow peak in the phase shift data. This is seen as the two-phase coexistence region. Also, in a continuous phase transition, due to the lack of latent heat and the growth of correlation length toward the macroscopic length-scales, there is a decrease in the phase shift data. In our studies we show the heat capacity and the simultaneously-taken phase shift data.

3.1.2 Thermometry.

Determining the temperature of the sample with a certain accuracy is the central priority of a thermodynamic measurement. To measure the temperature of the sample in our AC Calorimetry experiments, we used a resistive thermistor with a DC bias current.

The temperature is calculated by calibrating the thermometer as a function of resistance: T = *f*(R). Differentiating with respect to the temperature will give:

$$\frac{dT}{dR} = \frac{df(R)}{dR} = g(R) \Rightarrow dT = g(R)\,dR. \qquad (3.14)$$

Considering small temperature oscillations, the amplitude of these oscillations can be written as: $T_{ac} \cong dT = g\left(\overline{R}\right)dR$, and the average temperature about which the sample is oscillating is given by: $\overline{T} = f\left(\overline{R}\right)$, where $\overline{R}$ is the average resistance. Using the known relations between the voltage, current and resistance, we could write: $T_{ac} = g\left(\overline{R}\right)dR = g\left(\frac{V_{dc}}{I}\right)\frac{dV}{I} = g\left(\frac{V_{dc}}{I}\right)\frac{V_{ac}}{I}$, which gives the expression for the amplitude of the temperature oscillations in terms of the voltage drop across the thermometer. By increasing the current I, we can make the voltage as large as desired, and increase the power dissipated at the thermometer; but keeping it as low as possible to avoid self-heating, which reduces the accuracy in determining the absolute temperature.

In AC calorimetry, two types of thermistors were used: a platinum and a carbon flake thermistor. The platinum thermometer is used for regulation of the bath temperature, and the carbon flake thermistor is used for the cell's temperature. For the carbon flake thermistor, the calibration function is given by the following:

$$\frac{1}{T} = \sum_n A_n \log(R)^n = A_0 + A_1 \log(R) + A_2 \log(R)^2 + \ldots, \qquad (3.15)$$

for the platinum thermometer the expression will be given by:

$$T = \sum_n B_n R^n = B_0 + B_1 R + B_2 R^2 + \ldots, \qquad (3.16)$$

By using $T_{ac} \cong dT$, and differentiating Eq. (3.15), we can write the expression for the amplitude of the temperature oscillations [6] :

$$T_{ac} \cong dT = \frac{V_{ac}}{T^2 R_{th} I}\left[A_1 + 2A_2 \log\left(R_{th}\right) + 3A_3 \log\left(R_{th}\right)^2 + ... \right], \qquad (3.17)$$

where $R_{th} = \overline{R}$, $T = \overline{T}$, and V_{ac} is the amplitude of the voltage applied. The amplitude of the applied heat through a wire with resistance R using a voltage V, is given by:

$$Q_0 = \frac{V^2}{R} = \frac{1}{2} P_P, \qquad (3.18)$$

were P_P is the applied power. Using Eqs. (3.18), (3.17) and (3.10), we can get the expression for the heat capacity in terms of all experimentally measurable quantities:

$$C = \frac{P_P T^2 I R_{th}}{16\sqrt{2}\pi f}\left[A_1 + 2A_2 \log\left(R_{th}\right) + 3A_3 \log\left(R_{th}\right)^2 + ... \right]^{-1}. \qquad (3.19)$$

3.1.3 Design and operation of AC Calorimetry.

A block diagram of the design and electronic equipment of the AC Calorimeters used in this work are presented in Figure 3.4. At the very center of the setup stands the cell+sample+heater+thermometer system. This system is placed inside a cylindric copper block which provides for a very good temperature control and a controlled environment. The copper block temperature is controlled by a Lakeshore model 340 temperature controller via a proportional-integral-derivative (PID) feedback loop. At the rim of the block the temperature stability is ± 1 mK, while inside the temperature stability reaches on even 100 μK, because of the damping that comes from the massive copper block. Changing the setpoint of the temperature controller, we can achieve temperature scans

with different rates, which can vary from 10 mK/hr to 5 K/hr. Controlling the temperature of the bath is done by using a platinum PT-100 RTD (Resistive Temperature Device) very close to the cell.

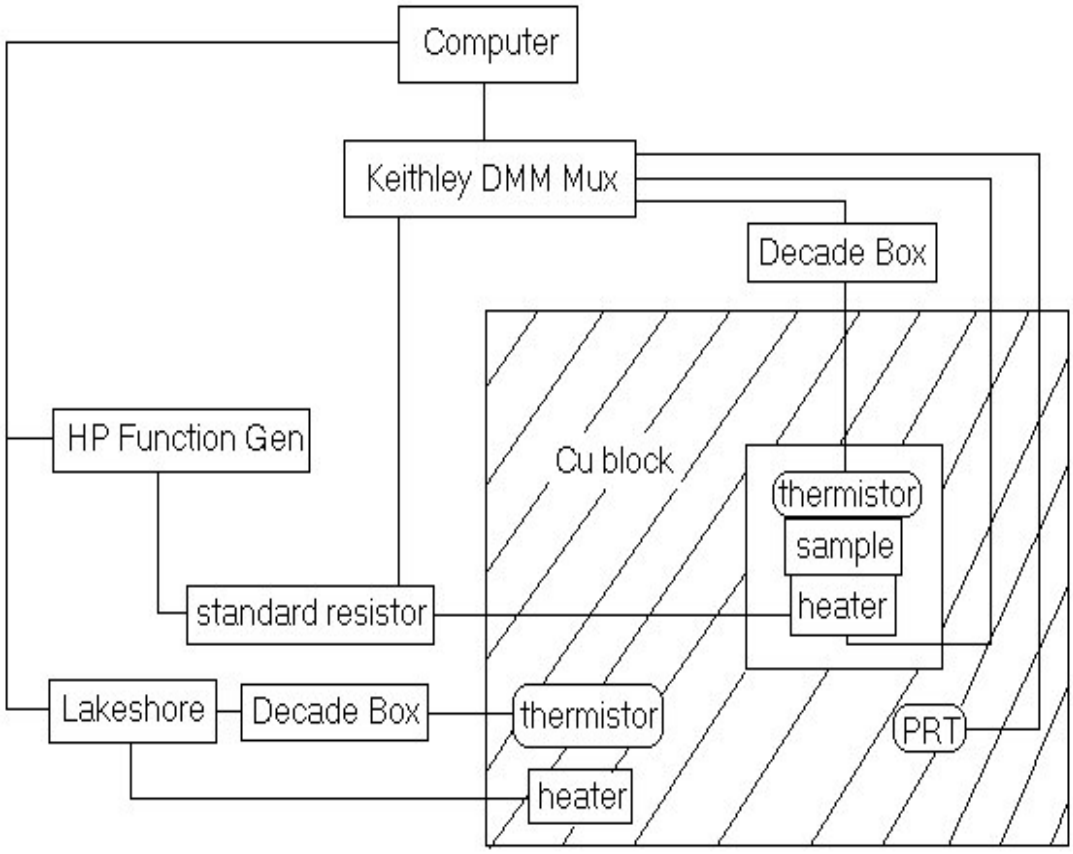


Figure 3.4: A block diagram of the AC Calorimeter design.

Monitoring the temperature oscillations of the sample is done using model 2002 Keithley Digital Multimeter. The Keithley DMM is used also to monitor the bath temperature, using another platinum thermometer. The power to heat the sample is provided sending a sinusoidal heating voltage through a strain-gauge from a Hewlett Packard, model 33120A, 15 MHz Function/Arbitrary Waveform Generator.

The data acquisition is achieved using software written in C++, compiled under Borland C++. The main thread achieves the data collection, and a secondary thread controls the temperature. The data collection runs as follows:

1. Measure the temperature of the bath before digitizing.
2. Digitize the temperature oscillations of the sample. Find the amplitude of the temperature oscillations Tac and the phase shift ϕ between temperature oscillations and the power by fitting the digitized waveform to a sum of sines and cosines.
3. Measure the temperature of the bath after digitizing and calculate the average temperature: $T_{ave} = \frac{T_{before\_digitizing} + T_{after\_digitizing}}{2}$.
4. Calculate the heat capacity and the phase shift using:

   $C^* = \frac{P_0}{\omega T_{ac}}$ and $\phi = \phi_{resistance} - \phi_{power}$.

5. Save to the data file (*.dat) the values of C*, T_{ave}, T_{ac}, ϕ, as well as the average resistance of the thermistor, R_{th}.

The temperature controlling thread runs as follow:

1. Start the loop that goes through all the zones.

2. Equilibrate the temperature at the starting point of the zone. Signals the main Digitizing thread.
3. Controls the temperature throughout the zone.
4. When the Digitizing is done, signals the main thread that the zone has ended and starts another zone.
5. Restart from the step 2 untill all the zones are done.
6. End of the main program.

3.2 NON ADIABATIC SCANNING CALORIMETRY

AC Calorimetry directly measures the heat capacity of the samples with a very high precision. However, this technique has the disadvantage in that it does not measure the enthalpy, so it cannot determine quantitatively. Non Adiabatic Scanning calorimetry has the ability to measure very small latent heats (in the range of mJ per gram), on very small samples. The method was used first in the mid ‘70s [7], and was presented as a very powerful method in estimating latent heat recently by Ema [8-10]. The experimental apparatus design used in non adiabatic scanning (NAS) calorimetry in our lab is actually a dual-mode calorimeter which can operate both in AC and NAS mode, using the same sample cell. Thus, we will not describe the design of the NAS calorimeter, as it is the same as the one described in the previous section for AC calorimeter, but will give a brief description of the theory of operation and the data acquisition process.

3.2.1 Theory of operation.

A very detailed and elegant description of the theory of operation for non adiabatic scanning (NAS) Calorimetry is given in [10].

The thermal model for NAS calorimeter is the model schematically given in Figure 3.1. The model is valid until the internal thermal diffusion time of the sample is small and the temperature gradients within the sample can be neglected. We discussed the fulfillments of these conditions in section 3.1.

Starting from the first law of thermodynamics, the energy given to a thermodynamic system (in our case cell+sample), for a constant pressure process can be written as:

$$dQ = Pdt = C_p^{sys} dT + dL + (T - T_B) K dt \quad \text{or} \quad P = C_p^{sys} \frac{dT}{dt} + \frac{dL}{dt} + (T - T_B) K , \qquad (3.20)$$

where $C_P^{SYS} = C_P^{CELL} + C_P^{SAMPLE}$, dL is the latent heat associated with the first order phase transitions, T is the sample temperature, T_B is the bath temperature, and K is the thermal conductance of the sample. Thus, outside the two-phase coexistence region, and no latent heat being released in the system, the heat capacity can be written as:

$$C_p^{sys} = \frac{P - (T - T_B) K}{\frac{dT}{dt}} . \qquad (3.21)$$

For a two-phase coexistence region, we can define an effective heat capacity of cell+sample system as:

$$C_{eff}^{sys} = \frac{P - (T - T_B)K}{\frac{dT}{dt}} \quad . \tag{3.22}$$

The latent heat released or absorbed from the system over a narrow region between temperatures T_a and T_b can be expressed as:

$$L = \int_{T_a}^{T_b} \left\{ C_{eff}^{sys} - \left[C_p(coex) + C_p(cell) \right] \right\} dT \, . \tag{3.23}$$

In the NAS mode, the DC heating power applied to the sample is linearly ramped and for a heating run can be expressed as:

$$\begin{cases} P = 0 \ldots\ldots\ldots\ldots\ldots\ldots t < 0 \\ P = \frac{dP}{dt} t \ldots\ldots\ldots\ldots 0 \le t \le t_1 \\ P = P_0 = \frac{dP}{dt} t_1 \ldots\ldots\ldots t > t_1 \end{cases} , \tag{3.24}$$

where $\frac{dP}{dt}$ is a constant.

The initial temperature of the sample is the one of the bath, T_B. For $t > t_1$, the power also will remain constant: $P = P_0 = \dot{P} t_1$, to allow the sample to equilibrate, untill it reaches a plateau temperature given by: $T(\infty) = T_B + \frac{P_0}{K}$. In a cooling run, the power profile will be reversed and will look like this:

$$\begin{cases} P = P_0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots t < 0 \\ P = P_0 + \frac{dP}{dt} t \ldots\ldots\ldots (\frac{dP}{dt} < 0) \ldots\ldots\ldots 0 \le t \le t_1 \\ P = 0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots t > t_1 \end{cases} . \tag{3.25}$$

In a cooling run the sample's temperature will initially have a $T(\infty)$ temperature, and then will decrease to a final value T_B, the bath's temperature. During the time interval from 0 to t_1, the sample's temperature will change as following:

$$\begin{cases} T(t) = T_B + \frac{1}{K}\dot{P}(t-\tau_{ext}) + \tau_{ext}\frac{1}{K}\dot{P}e^{-\frac{t}{\tau_{ext}}} \text{.....heating} \\ T(t) = T_\infty + \frac{1}{K}\dot{P}(t-\tau_{ext}) + \tau_{ext}\frac{1}{K}\dot{P}e^{-\frac{t}{\tau_{ext}}} \text{.....cooling} \end{cases}, \qquad (3.26)$$

where: $R = \frac{P_0}{T_\infty - T_B}$ is the thermal conductance of the sample. Using Eqs. (3.21) and (3.26), we can calculate the heat capacity for one-phase region from heating and cooling scans at any given temperature. In Eq. (3.21), P is the power at a given instant of time corresponding to a specific sample temperature T in the interval T_B to $T(\infty)$. The derivative *dT* / *dt* can be calculated by fitting the data from Eq. (3.26) over a short time interval centered at that time. We can determine the latent heat L directly from Eq. (3.23) for a firstly first-order phase transition [10].

3.2.2 Data acquisition process.

The acquisition of the data is automated and made using computer interfaces, with software written in C++, compiled under Borland C++. A linearly ramped power of the form "$P = P_0 t$" is sent to the cell by generating a voltage of the form $V = \sqrt{P_0 t}$ from Hewlett Packard, model 33120A, 15 MHz Function/Arbitrary Waveform Generator. The operation of the non adiabatic scanning mode of the calorimeter for a heating scan goes as follows:

1. The bath temperature is stabilized at the starting desired temperature.
2. The bath temperature and the sample temperature are measured by Keithley Digital Multimeter.
3. A linearly ramped power is being generated by the function generator. Simultaneously, the Keithley Digital Multimeter reads and monitors the temperature of the sample as a function of time $T(t)$.
4. A constant DC power P_0 is generated by the function generator. After a relaxation time of 900 sec., the bath temperature and sample temperature are recorded again by Keithley Digital Multimeter.
5. The power will be linearly ramped down from P_0 to zero by the function generator. Simultaneously the temperature of the sample is being monitored by Keithley DMM.
6. The function generator is turned off, so no power will be sent to the sample. After a relaxation time of 900 sec., Keithley DMM records the temperature of the sample and the bath.
7. The external thermal resistance of the sample is calculated using the expression: $R_e = \frac{P_0}{T - T_B}$.
8. The next setpoint for the bath temperature is sent to the Lakeshore. The temperature of the bath will be equilibrated.
9. Everything is repeated from the step 2, until the desired range of temperature scan is covered.
10. The effective heat capacity of the sample is calculated using the expression 3.22.

When a cooling scan is performed, every step is followed as above, with the difference that the scan sequence is reversed. First cooling then heating will be performed and then the bath temperature is set to a lower setpoint.

3.3 AC Calorimetry by Radio Frequency Heating.

Modulation AC Calorimetry technique described in 3.1 of this chapter, has the limitation that it relies on conduction to apply heat to the sample. The sample is put in contact with the heater, which will generate Joule heating by passing a voltage heating through it; but this will heat initially only the surface of the sample, and then through conduction the heat will be carried throughout the whole sample. This type of heating will suffer from the possibility of creating temperature gradients across the sample above certain operating frequencies of the applied power. In this work we describe a modified and improved version of the AC calorimetry technique using Radio Frequency Field Heating as the only source to apply heat to the sample. This type of calorimeter is the first of its kind, and we have the privilege to be the first ones to develop it among the research groups that use modulation calorimetry. In Chapter 2 we described the theory of Radio Frequency Heating. Being a different kind of AC Calorimeter, we will not describe the theory of operation, because we described that in section 3.1.1. In the following sections we will go over the design and operation of this new technique, pointing out the different specifics in the electronics aspects and data acquisition process.

3.3.1 Design of AC Calorimeter by Radio Frequency Heating.

A new calorimeter using Radio Frequency Heating called CalorRF, was homemade entirely in our lab. In the next sections we will describe the creation of the Radio Frequency Network as a very important part of the Radio Frequency Heating Process, cell design, and Temperature Control and Electronic Circuitry of the calorimeter.

3.3.1.1 Radio Frequency Heating Network.

The main elements of the Radio Frequency Heating Network are the energy source, the material to be heated, and a waveguide system to interconnect these two elements and provide a path for the energy from the source to the applicator. Sending Radio Frequency energy towards a material will have the following consequences: part of the energy will be absorbed by the material and part of it will be reflected off the material and is "wasted." To ensure and maximize the effective or absorbed part of the incident energy that ends up in the material, we can employ some form of matching or tuning network. The employment of the matching or tuning network becomes even more necessary knowing that the Radio Frequency electromagnetic waves have low energy, at least compared to the microwave energy (this is also the reason why Radio Frequency

ovens are not common in domestic use). This process of creating the proper tuning network is called matching or tuning the system to the load.

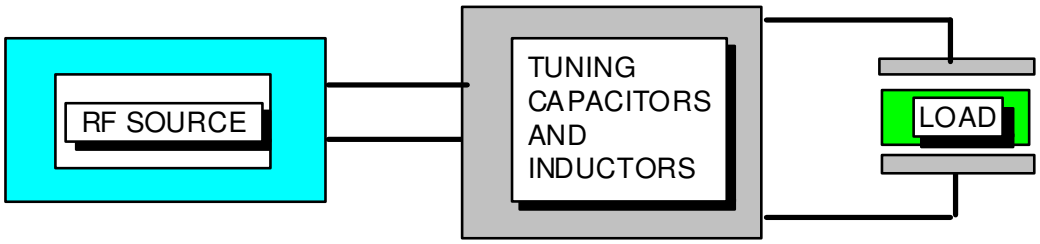


Figure 3.5: Radio Frequency Heating Network.

Figure 3.5 shows the schematics of a typical Radio Frequency Heating Network. The main parts of RF Network are: the Radio Frequency Energy source, the load to be heated and the Radio Frequency matching network, which is composed of capacitors and inductors. In our setup we are using and LRC series circuit as a matching network.

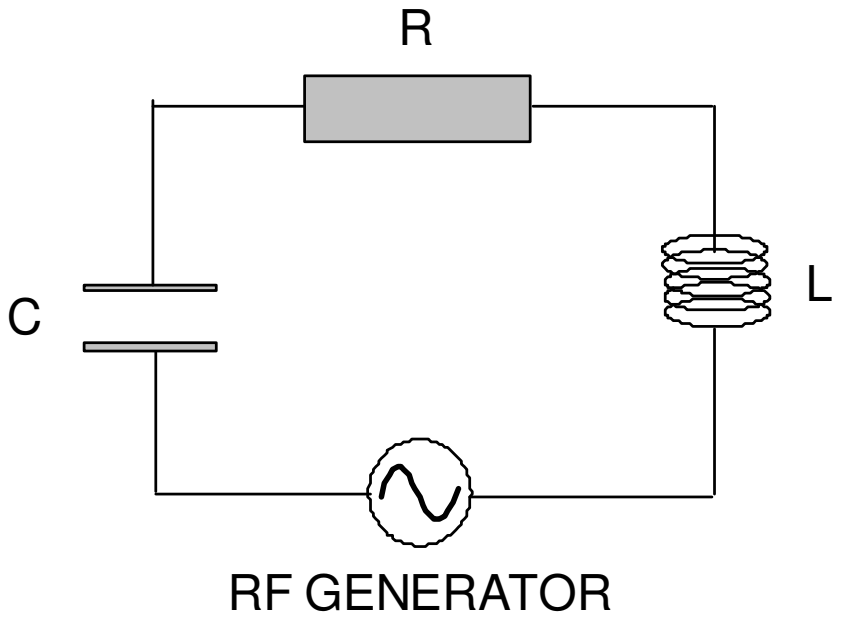


Figure 3.6: LRC series circuit used as radio frequency heating matching network.

Choosing the right values for the capacitance and inductance we can channel the natural frequency of the oscillations of this matching network to a particular convenient value. In the setup, the LRC series circuit has a natural frequency of f_{rf} = 1.13 MHz or an angular frequency $\omega_{rf} = 7.0964 \times 10^6$ rad/s. We generate the high frequency electromagnetic waves from the Model DS345 synthesized function generator. Our LRC matching network is connected to the cell inside where the sample to be heated is located, resulting in a lower natural frequency of oscillations of the whole matching-network+cell system, around $\omega_{rf} = 5.056 \times 10^6 \frac{rad}{s}$.

3.3.1.2 Cell Design.

With the Radio Frequency or dielectric heating as the only source of heating the sample, we create the possibility of eliminating the heater from the system whose heat capacity is to be measured. In the Radio Frequency Heating calorimetry technique, such a system is composed by the sample, thermometer, and the cell.

The cell holding the sample to be heated will serve as the capacitor. Between its plates we will send the high frequency alternative electric field. To serve this purpose, we choose to use the so-called Cup and Lid design for the cell.

In Figure 3.7 we present a schematics of Cup and Lid cell design used in our Radio Frequency heating calorimeter. For the cup and lid we use 99.95% pure, 0.1 mm in thickness silver foil, from Alfa Aesar [11]. The silver cup and lid are the plates of the capacitor, thus we need to separate them eliminating any electrical shorts using a 20x20 mm plastic sheet. The whole system, cup+lid+plastic-sheet+sample, is secured hermetically using plastic screws.

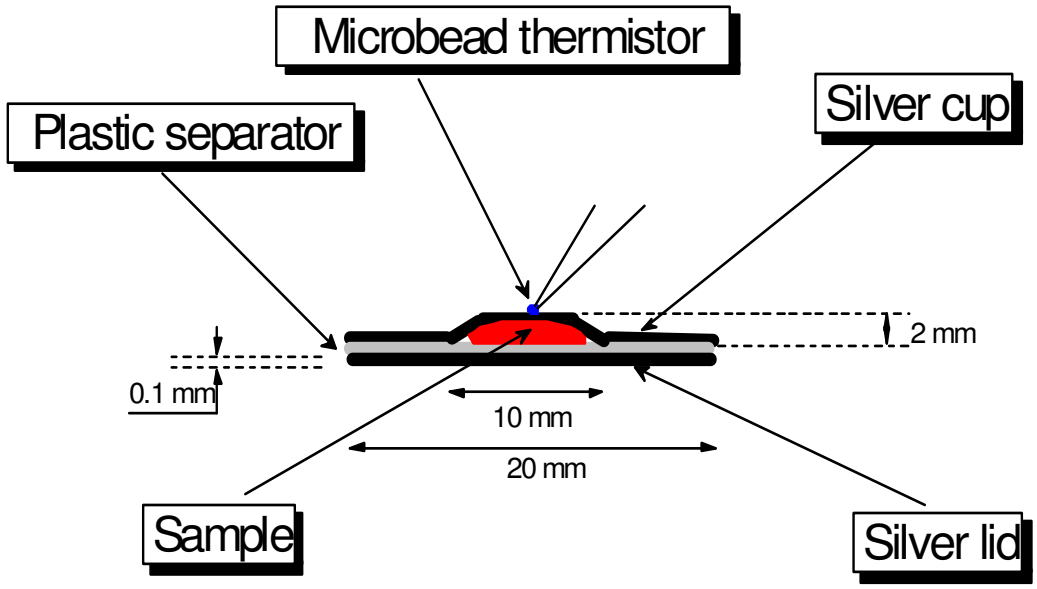


Figure 3.7: Cup and lid cell design used in the RF calorimeter.

The main advantages of the cup and lid design have been described in detail by other authors [12,13]. We can list some of these advantages which play a very important role in our results:

1. This design has a very good sensitivity response in measuring the heat capacity of liquid crystals.
2. It has a very good frequency scan response, a very important factor in minimizing the errors of the heat capacity measurements.
3. It has proved to be very easy in constructing the handling the cell compared to the other cell designs.

In Figure 3.7 we show also the temperature sensor, which is attached at the center of the cell, using a small drop of GE varnish. This temperature sensor is a Microbead thermistor, type 61A8, purchased from YSI [14]. The diameter of the sensor is about

0.254 mm, and its leads are around 8 mm long. These sensors have a time constant of around 0.5s, as shown by their manufacturer. They have a very small mass, less than 0.0001 mg, not contributing significantly to the cell+sample heat capacity.

3.3.1.3 Temperature Control

A very important part of Radio Frequency Heating Calorimeter as a modulation type calorimeter is the temperature control. The cell inside where the sample will be located will be placed inside a bath which will provide a very good temperature control. Holding the cell in its place will be the wires feeding the Radio Frequency alternating electric field, and the wires of the thermistor attached to the sample. In our setup, the bath is made out of a massive brass cylinder block. This brass block has walls 2.5 cm in thickness, and a cylindrical cavity 5 cm in height and 4 cm in diameter. A Kapton insulated flexible heater from Omega Enginnering is glued to the outside surface of the cylindrical brass block. Between the heater and the brass block surface we have attached a control thermometer, which is a platinum, PT-100, resistive temperature device (RTD). We connect the wires of the thermometer and of the heater with a Lakeshore model 340 temperature controller. By adjusting the PID settings of the Lakeshore, we can achieve very good temperature control of the bath in both scanning and stepping operational modes of the calorimeter. The brass block is place inside a cylindrical steel can, which is soaked 90% in a water bath, and providing a very good temperature controlled bath. At the rim of the brass cylinder the temperature stability is ± 1 mK over a 30 min time

interval, and around 100 μK inside the cavity where the cell+sample system is placed, because of the damping that comes from the massive brass block.

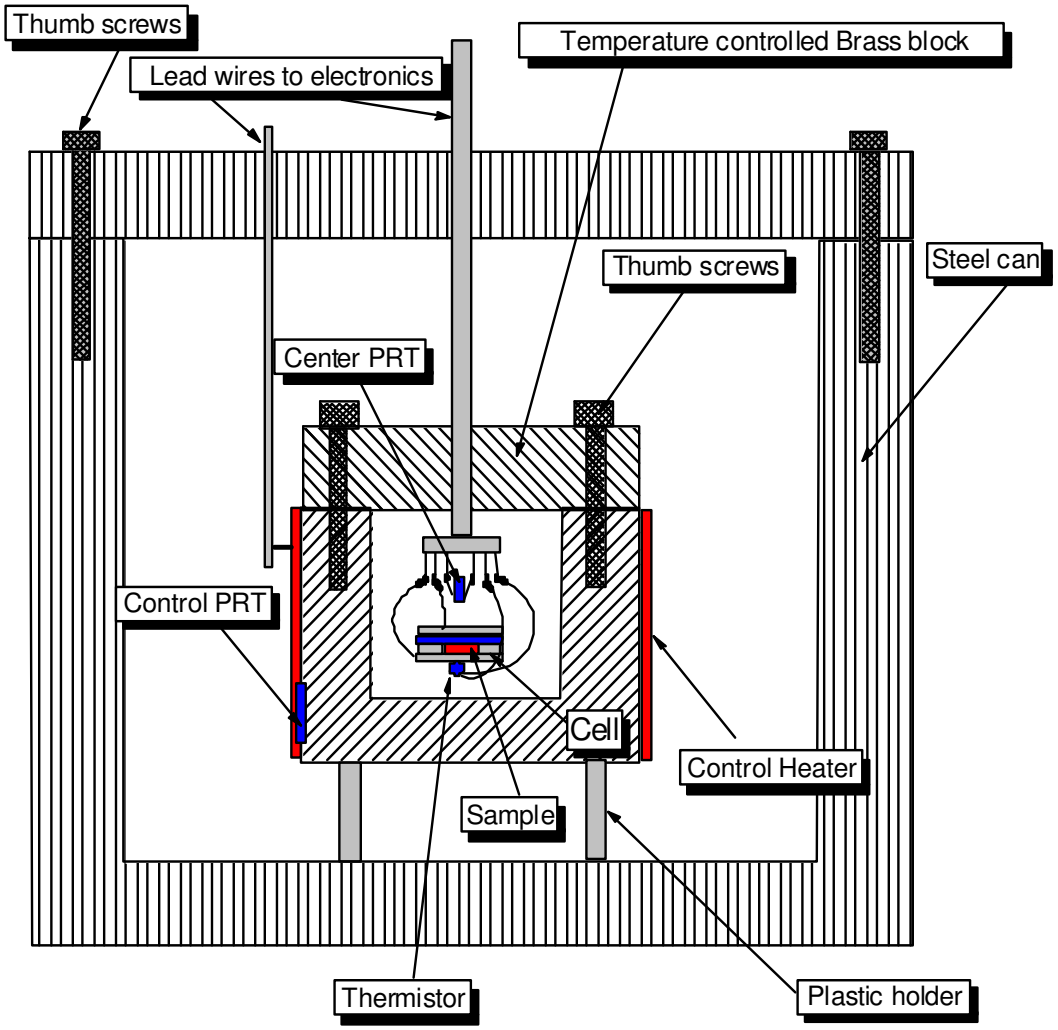


Figure 3.8: Design of Radio Frequency Calorimeter.

The bath temperature is monitored by using another PRT thermometer inside the cavity of the brass block, very close to the sample. This PRT is called the Center PRT.

3.3.1.4 Electronic Circuitry

In Figure 3.9 we show the block diagram of the electronics used for our Radio Frequency Calorimeter setup. A programmable arbitrary wave form generator model DS 345 from Stanford Research Systems [15], is used to supply the high frequency alternating electric field. It is connected through cables with the plates of the cell, which are the plates of the capacitor, inside where the sample is located. In our setup we use another programmable arbitrary wave form generator model DS335, from Stanford Research Systems, for two reasons:

1. As an Amplitude Modulated input, which controls the amplitude of the output waveform from function generator model DS345. Using this setup, we can modulate the amplitude of the high frequency alternating electric field with the amplitude modulation frequency typical for AC calorimetry (~15 mHz).
2. As a TTL Trigger for the Digital Multimeter to start digitizing the resistance of the thermistor.

The Radio Frequency Heating Matching Network is represented in figure 3.9 by the LRC-series circuit, which is connected between the function generator DS345 and the plates of the cell through cables.

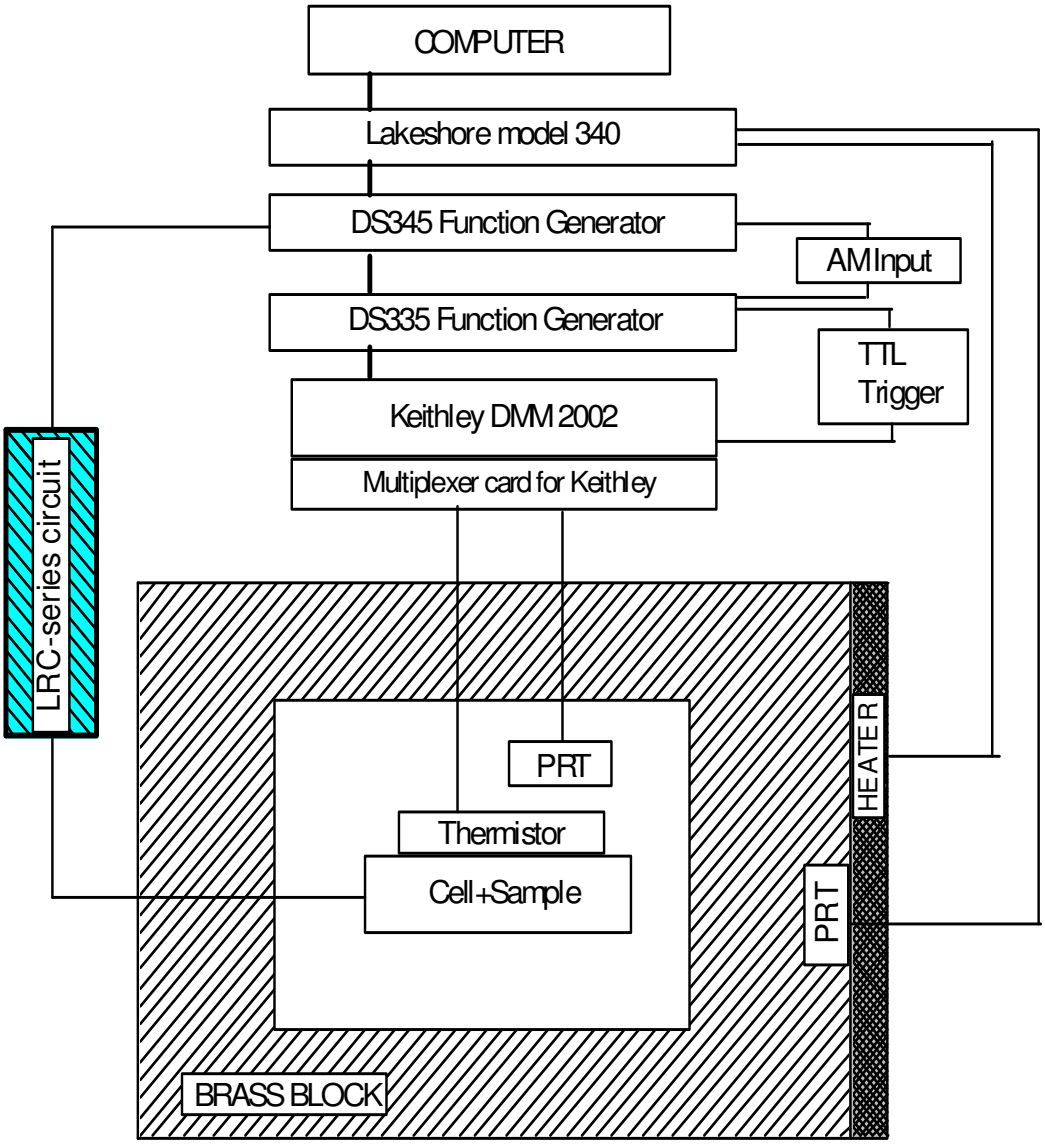


Figure 3.9: Block diagram of the electronics for the Radio Frequency Calorimeter.

The heater wrapped around the brass block and PRT thermometer controlling the temperature at the rim of the brass block, are connected through cables with the Lakeshore model 340 temperature controller [16].

Keithley Model 2002 Digital Multimeter [17], is connected with the center PRT inside the cavity of the brass block, very close to the sample, and the thermistor attached to the sample through cables, to monitor the temperature of the bath and the sample. A multiplexer scanner card model 2001-SCAN [18] is connected to Keithley to better use its capabilities of measuring different resistances through different channels.

The DS345 Function Generator, DS335 Function Generator, Keithley DMM, and Lakeshore temperature controller are all connected with the Pentium PC operating under Windows 98, through IEEE-488 cables bought from [19].

3.3.2 Data Acquisition Process.

For the Radio Frequency Heating Calorimeter, the data acquisition program is written in C++ and compiled under Borland C++. In building the program, special attention was dedicated to the fact that the data collection and temperature control routines should be done simultaneously, thus, the software was written in multithreaded fashion; one main thread which does the data collection, the calculations and the printing of the results to the files, and a secondary thread controls the temperature. All the initialization parameters such as the temperature scan zones, the temperature scan ramping rates, the frequency of operation, the amplitude of the high frequency alternating electric field, the file names, the temperature coefficients for the thermistor to convert the

measured resistance into temperature, are easily modified to the users desire, by modifying the configuration file "RF.ini".

The main thread of the data acquisition program goes as follows:

1. The function generator DS345 is instructed to output the high frequency (~850 kHz) sinusoidal electric field with a certain desired amplitude.
2. The function generator DS335 is instructed to output the low frequency (~15 mHz) sinusoidal electric field with a certain desired amplitude. This will make the DS345 generate the desired Amplitude Modulated high frequency alternating electric field.
3. The program checks with the temperature control thread to see if the desired temperature is reached. If yes, go to the next step. If no, wait for it.
4. Keithley DMM will record the bath temperature before Digitizing T_b.
5. The digitization of the thermistor's resistance will start, done by Keithley DMM for 10 voltage periods (~66.6 sec.).
6. The fitting routine of the waveform received will start, recording the amplitude of the resistance and the phase shift between the power applied and the resistance oscillations. Then the program converts these to the amplitude of the temperature oscillations and the phase shift of the temperature. Also from the fit, we can record the average thermistor resistance.
7. Keithley DMM will record the bath temperature T_a after Digitizing.

8. The program will calculate the heat capacity C* using 3.10, the phase shift: $\phi = \phi_{resistance} - \phi_{power}$, and the bath temperature as: $T_{PRT} = \frac{T_b + T_a}{2}$.
9. The program will save all the calculated data to an output file.
10. The program checks to see if the temperature control thread signals the end of the temperature scan. If no, everything will be repeated starting from step 4. If yes, the program will signal the end.

The temperature controlling thread runs as follow:

1. Starts the loop that goes through all the zones.
2. Equilibrates the temperature at the starting point of the zone. Signals the main Digitizing thread.
3. Controls the temperature throughout the zone.
4. When the Digitizing is done, signals the main thread that the zone has ended and starts another zone.
5. Restart from the step 2 untill all the zones are done.
6. End of the main program.

3.4 SAMPLE CHARACTERISTICS.

3.4.1 Aerosil Gel.

The aerosil type 300 consists of SiO_2 (silica) spheres of diameter 7 nm, coated with OH-hydroxyl group. Because of the existence of hydroxyl groups on the surface of the aerosil spheres, hydrogen bonding will occur between aerosil particles. Dispersing aerosil particles in organic liquid medium will make them attach to each other, forming a gel by diffusion-limited aggregation. The specific surface of the type 300 aerosil is around 300 m^2/g.

Figure 3.10 shows a cartoon of aerosil gel formed by dissolving aerosil particles into octylcyanobiphenyl (8CB) liquid crystal, which we will describe in the next section. As Figure 3.10 shows, randomly crossing "pearl necklace" of silica are formed. A typical gel formed this way is generally called thixotropic, because of their ability to break easily, caused by the weak nature of hydrogen bonded silica chains. Liquid Crystals incorporating low-density gels formed by silica aerosil dispersions have been shown to be good experimental models to investigate both the isotropic to nematic (I-N), and the nematic to smectic-A (N-SmA), phase transitions in the presence of the static (quenched) positional and orientational randomness provided by surface coupling at the silica-LC interfaces [20]. The gelation threshold for the aerosil is for a density of $\rho \sim 0.015$ g/cm^3.

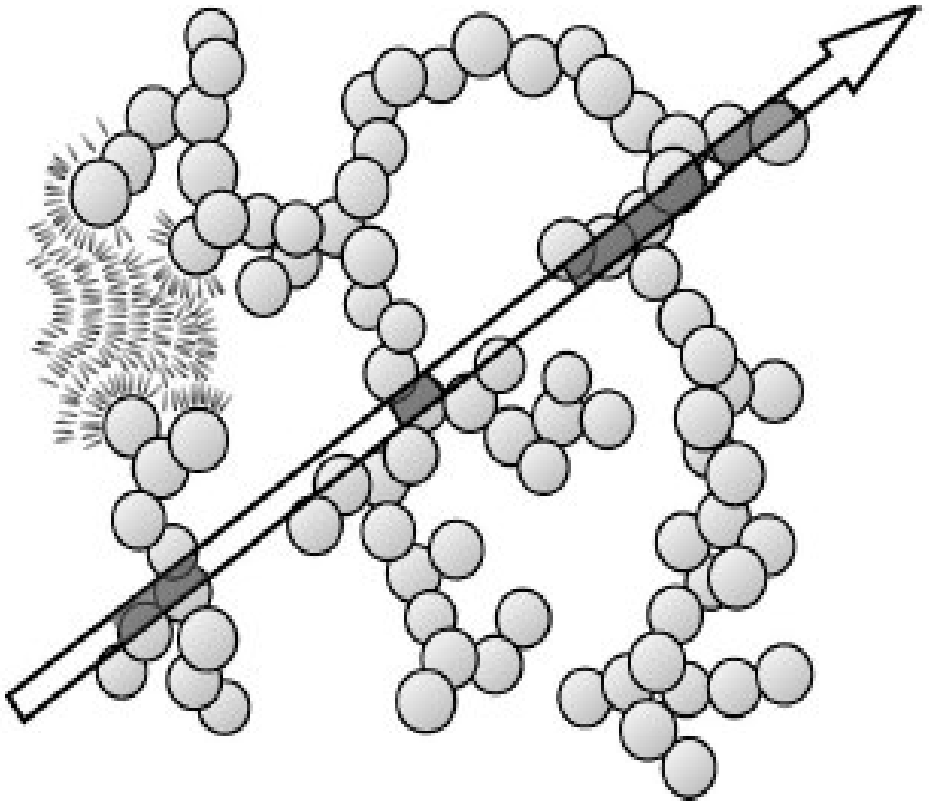


Figure 3.10: Cartoon of aerosil gel formed by diffusion limited aggregation process. Circles represent aerosil particles, “hairs” in the upper left corner represent 8CB molecules. Open and shaded parts of the arrow show void and solid chords, respectively.

3.4.2 Characteristics of 8CB.

Octylcyanobiphenyl, known as 8CB, is one of the most studied Liquid Crystal molecules. Figure 3.11 shows the molecular structure and Figure 3.11 shows the schematics of the molecule.

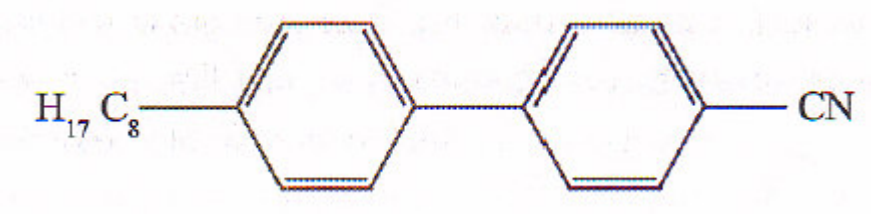


Figure 3.11: The molecular structure of the 8CB molecule.

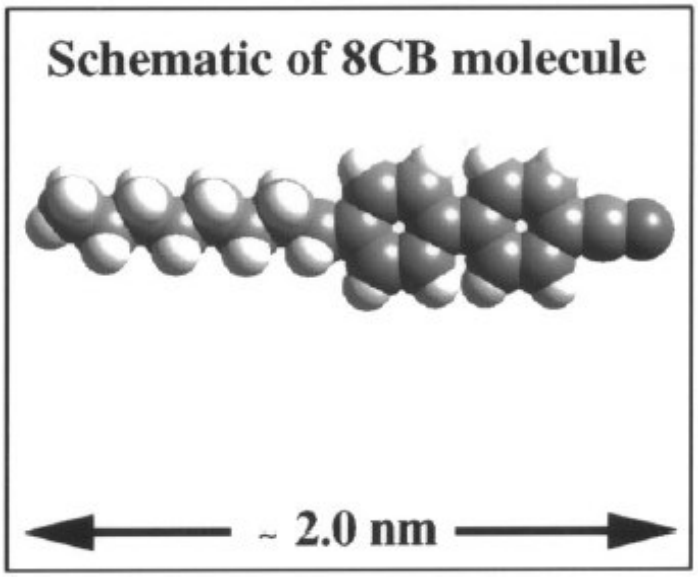
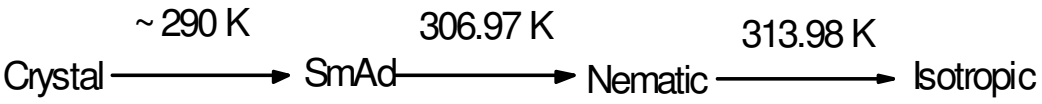


Figure 3.12: Schematics of the 8CB molecule.

Shown in figure 3.11, 8CB has a rod-like molecule, with a rigid biphenyl core, to which are attached an aliphatic tail and a polar cyano-head group. The molecular weight of

8CB is 291.44 g/mol. The phase transitions that the pure 8CB undergoes are shown below:



3.4.3 Characteristics of CCN47.

Figure 3.13 shows the molecular structure of CCN47, which is the abbreviated name of the liquid crystal 4'-transbutyl-4-cyano-4-heptyl-bicyclohexane.

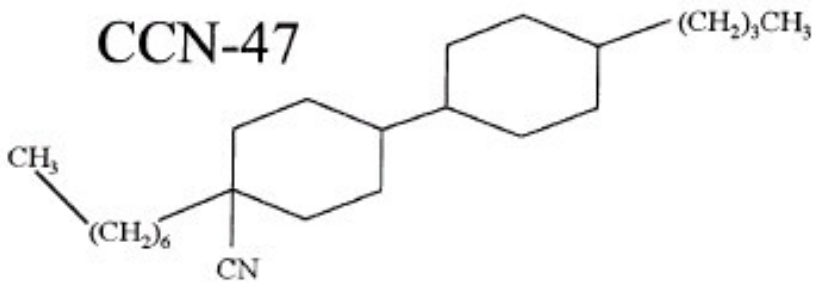
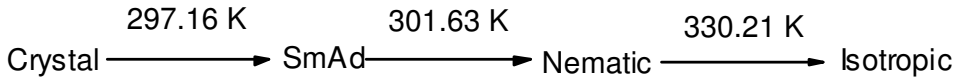


Figure 3.13: The molecular structure of CCN47.

In CCN47 molecule, the biphenyl group has been replaced by saturated hydrocarbon rings, making CCN47 a liquid crystal with a low birefringence, about 1/10 that of

cyanobiphenyls. The CCN47 liquid crystal features a wide range nematic phase, and undergoes the following transition:



The molecular weight of CCN47 is 345.617 g/mol.

References:

[1] Paul F. Sullivan, G.Seidel, Phys. Rev. 173. 679. (1968).

[2] H. S. Carslaw and J. C. Jaeger, Conduction of Heat in Solids, (Oxford, 1959).

[3] C. W. Garland, Thermchim. Acta 88, 127 (1985).

[4] X. Wen, C. W. Garland and M. D. Wand, Phys. Rev. A42, 6087 (1990).

[5] C. W. Garland, in Phase Transitions in Liquid Crystals, edited by S. Martelluci and A. N. Garland, (Plenum Press, 1992).

[6] D. Finotello, S. Qian, G. S. Iannacchione, Thermochimica Acta 304/305 (1967) 303-316.

[7] D. Djurek, J. Baturic-Rubcic, and K. Franulovic, Phys. Rev. Lett. 33, 1126 (1974).

[8] K. Ema, T. Uematsu, A. Sugata, and H. Yao, Japan, J. Appl. Phys., Part 1, 32, 1846 (1993).

[9] K. Ema, H. Yao, I. Kawamura, T. Chan, and C. W. Garland, Phys. Rev. E 47, 1203 (1993).

[10] H. Yao, K. Ema, C. W. Garland, Rev. Sci. Inst. 69 (1), 172 (1998).

[11] AlfaAesar, 30 Bond Street, Ward Hill, MA 01835, USA.

[12] K. Stine, Thesis in Chemistry, PhD thesis, M. I. T., 1988.

[13] T. Chan, Smectic Phase Transitions in Chiral Liquid Crystals. PhD thesis, M. I. T., 1995.

[14] YSI Temperature, 118 Victory Road, Springfield, NJ 07081, USA.

[15] Stanford Research Systems, 1290-D Reamwood Avenue, Sunnyvale, California 94089.

[16] Lake Shore Cryotronics, Inc., 575 McCorkle Blvd., Westerville, Ohio 43082, USA.

[17] Keithley Instruments, Inc., 28775 Aurora Road, Cleveland, OH 44139, USA.

[18] Keithley Instruments, Inc., http://www.keithley.com.

[19] L-com, Inc. 45 Beechwood Drive, North Andover, MA 01845, USA.

[20] T. Bellini, Phys. Rev. E 57, 2996 (1998).

CHAPTER 4

RADIO FREQUENCY HEATING CALORIMETRY RESULTS

4.1 Initial results in testing the Radio Frequency Heating calorimeter.

In this chapter we will describe first the preliminary results in testing the functionality of the new, first of its kind, completely homemade in our lab for the first time anywhere, AC Calorimeter by Radio Frequency Heating, as well as the use of the calorimeter in studying phase transitions in 8CB+aerosil dispersions. Traditional AC calorimetry technique uses Joule's heating as the only source to heat the sample. The sample is put in contact with the heater, which has been fed an alternating heating voltage. In Radio Frequency Heating calorimetry technique, the sample to be heated is placed inside the plates of the capacitor, where it becomes the dielectric of the capacitor. A high frequency electromagnetic field causes the molecules of the dielectric to distort and align millions of times per second, causing the heating of the sample. In Chapter 2 we described the theoretical aspects of radio frequency heating, and in Chapter 3, the design and process of setting up the calorimeter. In the next sections we will be showing data demonstrating the ability of the Radio Frequency Heating to be used for modulation

AC calorimetry: the detected temperature rise of the sample using the high frequency electric field power, the modulation of this power with frequencies typical for modulation AC calorimetry, the resulting temperature response, the frequency scans, and the temperature scan results. These preliminary tests were done using 8CB+aerosil dispersion samples, at a silica density of 0.1 g/cm^3.

4.1.1 Temperature rise.

In Chapter 2 we obtained the final Eq. (2.22) for the power absorbed by the sample inside the capacitor plates driven by a voltage V_0, at a driven frequency ω_d:

$$P = \frac{A}{d}\omega_d \left(\tan\delta\right)\left|\varepsilon\right| \frac{V_0^{\,2}}{\left(1-\left(\frac{\omega_d}{\omega_0}\right)^2\right)^2 + \left(\frac{\omega_d}{Q\omega_0}\right)^2}. \tag{4.1}$$

In Chapter 3 we described the design of the Radio Frequency Heating Calorimeter. If we consider the external thermal conductance K_e, between the cell + sample + thermometer system and the bath, as defined in Chapter 3, then the final temperature rise in the sample placed between the plates of the capacitor driven by a voltage V_0, at a driven frequency ω_d, will be given by:

$$\Delta T = \frac{P(\omega_d)}{K_e} = \frac{A}{dK_e}\omega_d \left(\tan\delta\right)\left|\varepsilon\right| \frac{V_0^{\,2}}{\left(1-\left(\frac{\omega_d}{\omega_0}\right)^2\right)^2 + \left(\frac{\omega_d}{Q\omega_0}\right)^2}, \tag{4.2}$$

where ω_0 is the natural frequency of oscillations of LRC circuit series, used as the matching network for radio frequency heating technique, and given by:

$$\omega_0 = \frac{1}{\sqrt{LC}}. \tag{4.3}$$

When we drive the LRC series circuit at the resonance frequency $\omega_d = \omega_0$, the temperature rise in the sample inside the plates of the capacitor will be maximum and according to 2.28 can be written as:

$$\Delta T = \frac{A}{dK_0}\omega_0 \frac{\varepsilon_2}{\varepsilon_1}|\varepsilon| Q^2 V_0^{\,2} \tag{4.4}$$

We have performed our initial tests on a octylcyanobiphenyl (8CB) + 70-$\overset{0}{A}$-diameter silica spheres (aerosil) sample with a density of ρ_S = 0.1 g/cm^3. Experimentally, we can see the temperature rise given by Eq. (4.4) by changing the driving frequency of the LRC series circuit, sending sinusoidal voltages at different frequencies from the DS345 function generator between the plates of the capacitor, but keeping all the other parameters: the amplitude of the voltage V_0, the specifics of the cell, *A* area and *d* the distance between capacitor plates, constant. Figure 4.1 shows the change in cell + sample temperature as a function of the frequency of the applied field conducted at a fixed temperature of 310 K. To generate this data, the function generator outputs a sine wave at different frequencies between 1 and 14 MHz, with and amplitude of 10 Vpp. Approaching the natural frequency of oscillations of the matching network + cell + sample system, f_{rf} = 0.805 MHz or $\omega_{rf} = 5.056\times10^6 \frac{rad}{s}$, the voltage drop across the plates of the cell will increase, having a peak at f_{rf} = 0.805 MHz, and decreasing after that. According to Eq. (4.4), the temperature rise in the sample will follow an increase, a peak and then a decrease, perfectly seen in our results shown in Figure 4.1.

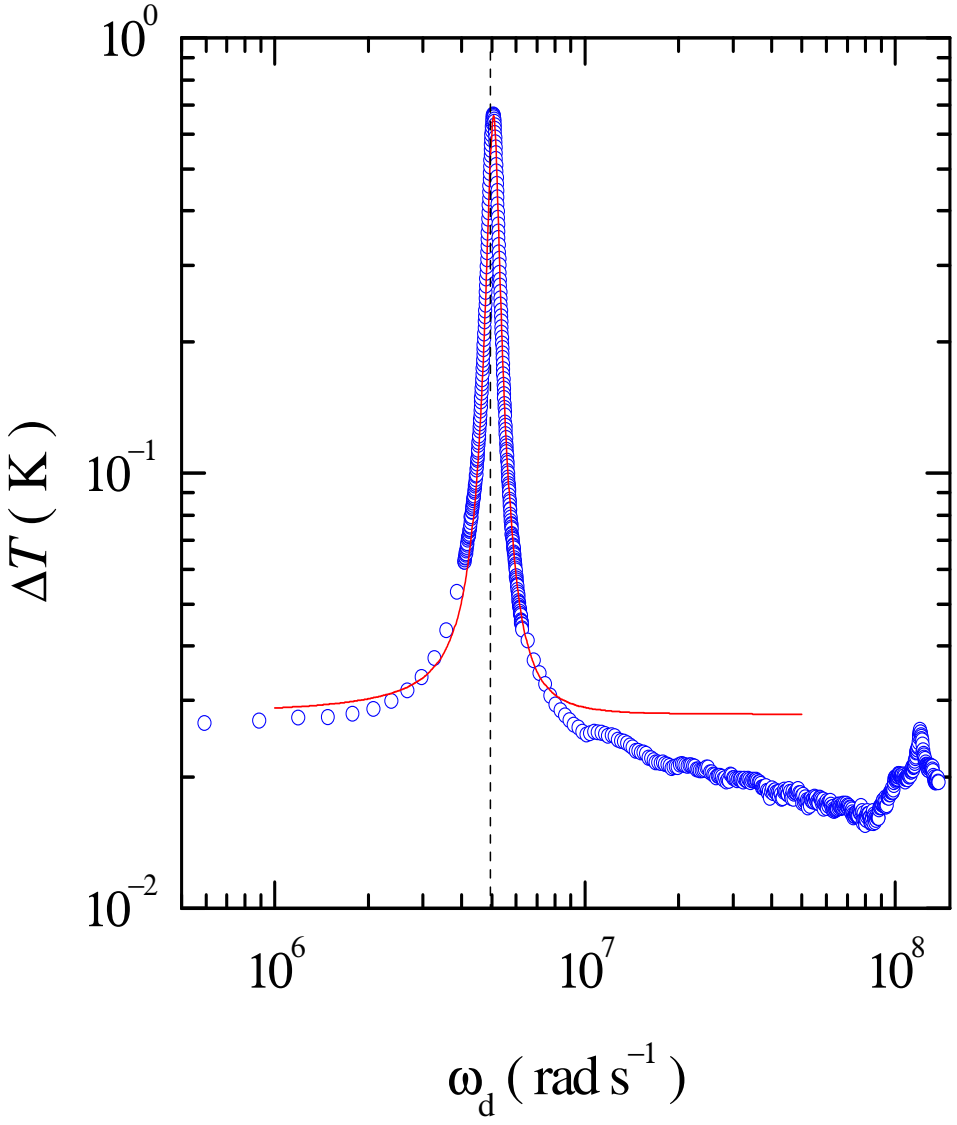


Figure 4.1: The change in temperature of the cell + sample system as a function of the frequency applied field conducted at a base temperature of 310 K. Solid line is a fit using Eq. (4.4) and assuming a constant $|\varepsilon|$ and $\varepsilon_2 / \varepsilon_1$.

The solid line in Figure 4.1 represents a driven damped oscillator fit model giving a driving amplitude of $A_0 = 8.4\times10^{-10} mK$ and a quality factor of the circuit of Q = 12. As we can see from Figure 4.1, the quality of the fitting is very good at lower frequencies, perfect around the temperature rise peak, and then deviated from the data at higher frequencies. The reason for this deviation of the fit from the data at higher frequencies is that our fitting model considers only the driven damped oscillator model, not the frequency dependence of permittivity of the sample. According to Eq. (4.4), the temperature rise in the sample depends on the total permittivity of the dielectric material which is placed inside the cell, and the imaginary and real parts of the permittivity. This dependence becomes very important at higher frequencies as Figure 4.1 shows.

Also shown in Figure 4.1 is a very small peak of around 8 mK, at around 12 MHz. This very small response temperature rise of the sample, which highlights the very high resolution capabilities of our brand new radio frequency calorimeter, occurs as a result of excitation of other modes of the dielectric molecules in the material.

We conducted the same exact experiment with no sample inside the cell, and no features were seen, indicating that the temperature rise in the sample occurs only because of the radio frequency or dielectric heating of the material.

The data shown in Figure 4.1 represents the raw temperature rise in the sample, without any averaging or smoothing routine used, which highlights again the very high-resolution capabilities of our calorimeter.

4.1.2 Frequency scan results.

The basics of the AC calorimetry technique consist of applying periodically modulated sinusoidal power to the material to be studied, and monitoring the resulting temperature response of the material, which will be sinusoidal oscillations. We discussed the theory of modulation calorimetry method in Chapter 3. In order to successfully use Radio Frequency Heating for AC calorimetry, we need to amplitude modulate the high frequency alternating electric power being sent to the sample, with the amplitude modulation frequencies typical for AC calorimetry technique.

In amplitude modulation, the circuit or the modulator combines the carrier wave shown in Figure 4.2a, and the message signal shown in Figure 4.2b, to form a modulated wave shown in Figure 4.2c, that is a carrier wave with change in amplitude.

In our radio frequency calorimeter setup, as described in details in Chapter 3, section 3.3.1.4, we generate the carrier wave high frequency alternating electric field, using programmable arbitrary wave form generator model DS345. The frequency of this wave is f = 0.805 MHz. We generate the AM modulation waveform used to AM modulate the high frequency alternating electric field, using programmable arbitrary wave form generator model DS335. The frequency of this wave, for the purpose of seeing the temperature oscillations in the sample, is f = 15 mHz, corresponding to a period of oscillations of 66.6 sec. We chose this frequency as being a typical frequency at which we generally operate the other traditional AC calorimeters in our lab.

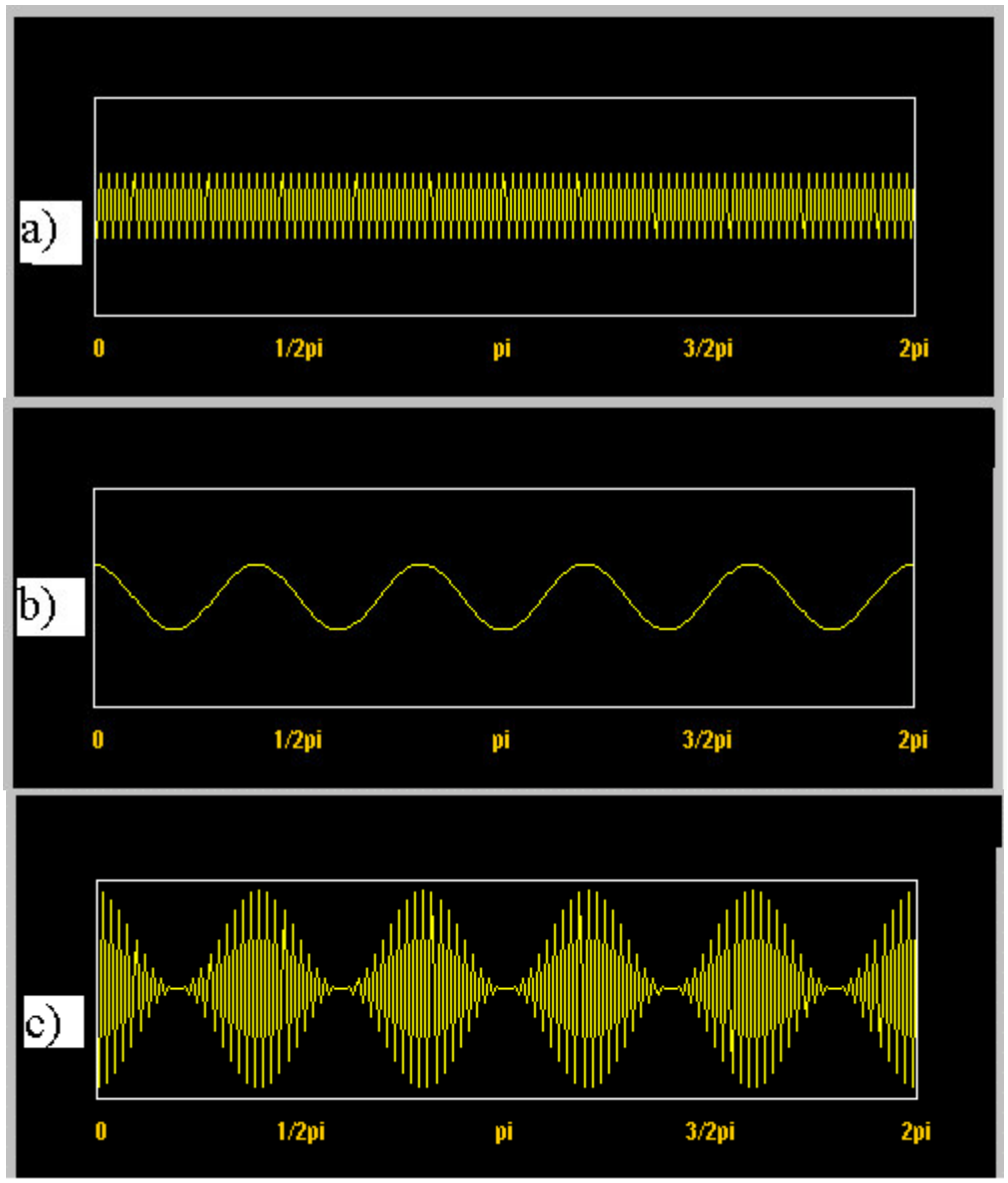


Figure 4.2: a) Schematics of the carrier wave high frequency alternating electric field used in radio frequency calorimeter; b) Schematics of the am modulation waveform used to modulate the high frequency alternating electric field. c) Schematics of the final modulated high frequency alternating electric field used in radio frequency calorimeter.

Combining these two signals will result in the creation of the final AM modulated high frequency alternating electric field as shown in Figure 4.2c used in radio frequency calorimeter, which is generated from the arbitrary wave form generator model DS345. This final AM modulated high frequency alternating electric field shown in Figure 4.2c has an amplitude of 10 Vpp, and a modulation depth of 100%. Thus the amplitude will change from 5 to –5 V, at a frequency f = 15 mHz or with a period of 66.6 sec.

After sending the final AM modulated high frequency alternating electric field between the plates of the capacitor where the sample is located, as described above, we monitored the response temperature oscillations T_{AC} in the sample. Such temperature oscillations are presented in Figure 4.3. The relatively large T_{AC} response of more than 0.6 K, comes from the fact that we have applied a high frequency alternative electric field with a relatively large voltage amplitude of 10 Vpp, modulated in amplitude as described above. The period of temperature oscillations of 66.6 sec., shown in Figure 4.3, perfectly matches the period of the applied AM modulated alternating electric field of 66.6 sec. This indicates the direct heating by radio frequency or dielectric heating of the sample, and not conductive or Joule's heating, which has a response temperature oscillation period half of the applied voltage heating.

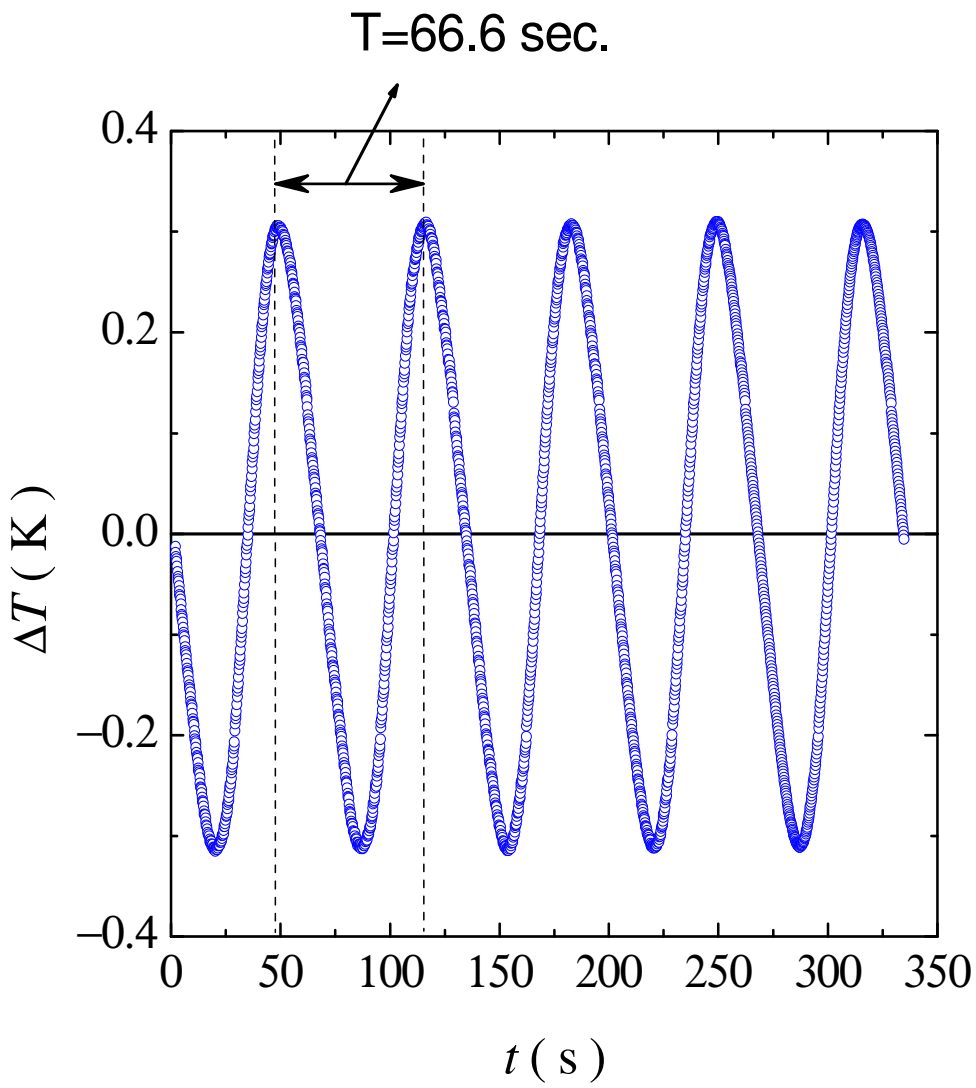


Figure 4.3: Sample temperature oscillations achieved by modulating the amplitude of the base frequency voltage at f_{rf} = 0.805 MHz, with an AM modulating frequency f_{am} = 15 mHz.

We repeated the same experiment with no sample inside the capacitor plates, and no temperature oscillation response was seen.

The data shown in Figure 4.1 represents the raw temperature oscillations in the sample, without any averaging or smoothing routine used, which highlights again the very high - resolution capabilities of our calorimeter.

The heat capacity of the sample which exhibits temperature oscillations with an angular frequency ω, as discussed in Chapter 3, is inversely proportional to the magnitude T_{ac} of the induced temperature oscillations. According to Eq. (3.10) it is given by:

$$C \cong C^* = \frac{\dot{Q}_0}{2\omega T_{ac}}, \qquad (4.5)$$

where C is the heat capacity of the cell + sample + thermistor system, $\dot{Q}_0$ is the amplitude of the applied power to this system, $\omega = 2\pi f$ is the angular frequency of the applied power as well as the induced temperature oscillations, and T_{ac} is the amplitude of these oscillations.

The approximation shown in Eq. (4.5) for the heat capacity holds for certain requirements and is shown in Chapter 3, section 3.1.1 to stand for those ω angular frequencies determined from the so called frequency scans. A frequency scan determines the frequency regime over which the measured heat capacity will be frequency – independent.

We perform this frequency scan keeping the same base frequency of $\omega_{rf} = 5.056$ Mrad/s, or $f_{rf} = 0.805 MHz$, the same voltage, the same bath temperature, while

sweeping the modulation frequency and measuring the resulting temperature oscillations of the sample.

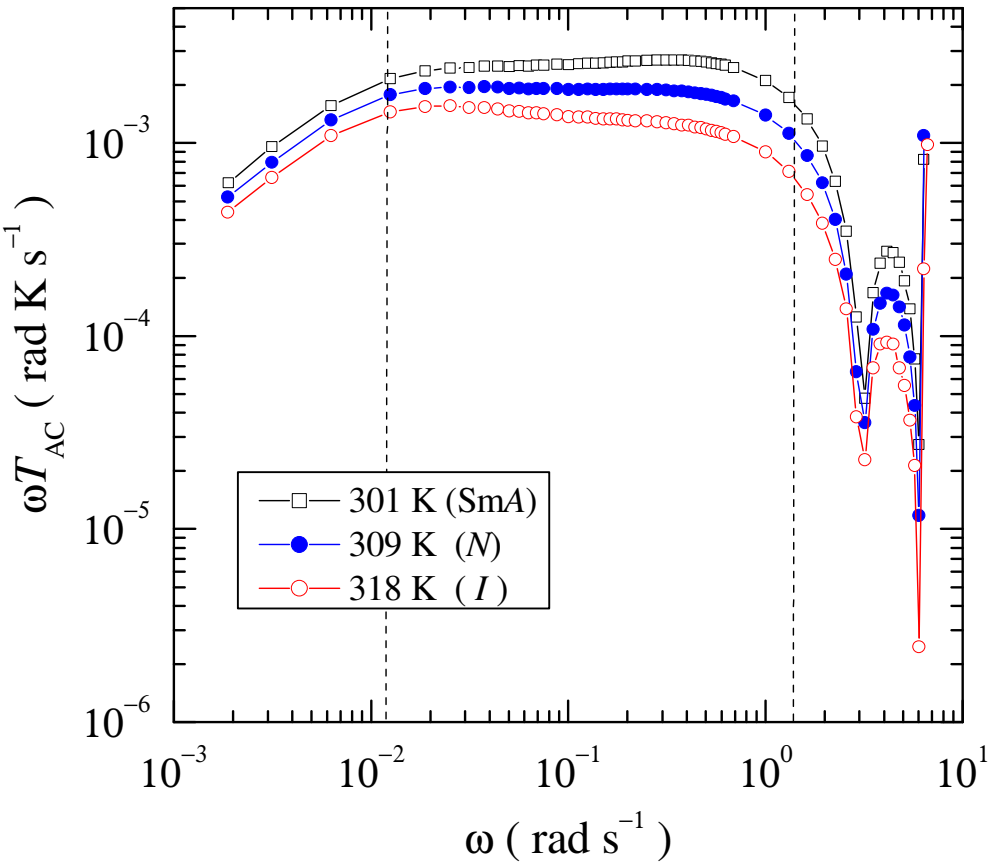


Figure 4.4: Typical frequency scans for 8CB+aerosil sample at 0.1 g/cm^3 silica density taken by the RF/AC calorimeter at three different temperatures corresponding to the isotropic (*I*), nematic (*N*), and Smectic-A (*SmA*) phases.

Figure 4.4 shows data from 3 frequency scans performed at the temperatures, 301, 309, and 318 K, corresponding to three different states of the 8CB + aerosil sample at 0.1 g/cm^3 silica densities, smectic-A, nematic and isotropic respectively. From the above plots we clearly see a wide plateau, which indicates the region where the heat capacity of the sample can be expressed by Eq. (4.5) and is frequency independent. A typical mass of cell + sample used in AC calorimetry in previous applications is around 0.2 g. In the present setup, the mass of the cell + sample system is more than 2 g. For such a relatively large mass, a very narrow plateau, or region for AC calorimetry operating frequencies, would be expected. Easily noted in Figure 4.4 is a very large plateau in frequency scans taken with our setup, making it very comfortable for us to choose among a wide range of operating frequencies. This wide range of operating frequencies can be found in the plot above between the so – called roll – off frequencies, represented by the dash lines in Figure 4.4. These roll – off frequencies can be used to find the internal and external time constants for the sample.

The very good frequency response of our radio frequency heating calorimeter setup comes from the advantage of heating the sample using radio frequency field heating, direct result of body (volume) heating.

Alternatively, we can plot T_{ac} vs. the frequency ω. A typical plot for the 8CB + aerosil sample at 0.1 g/cm^3 silica density, at three different temperatures is shown in Figure 4.5. As the frequency approaches zero, the resultant amplitude temperature oscillations T_{ac}, gets bigger, approaching the DC part of the heating, T_{DC}, which comes from the *rms* heating.

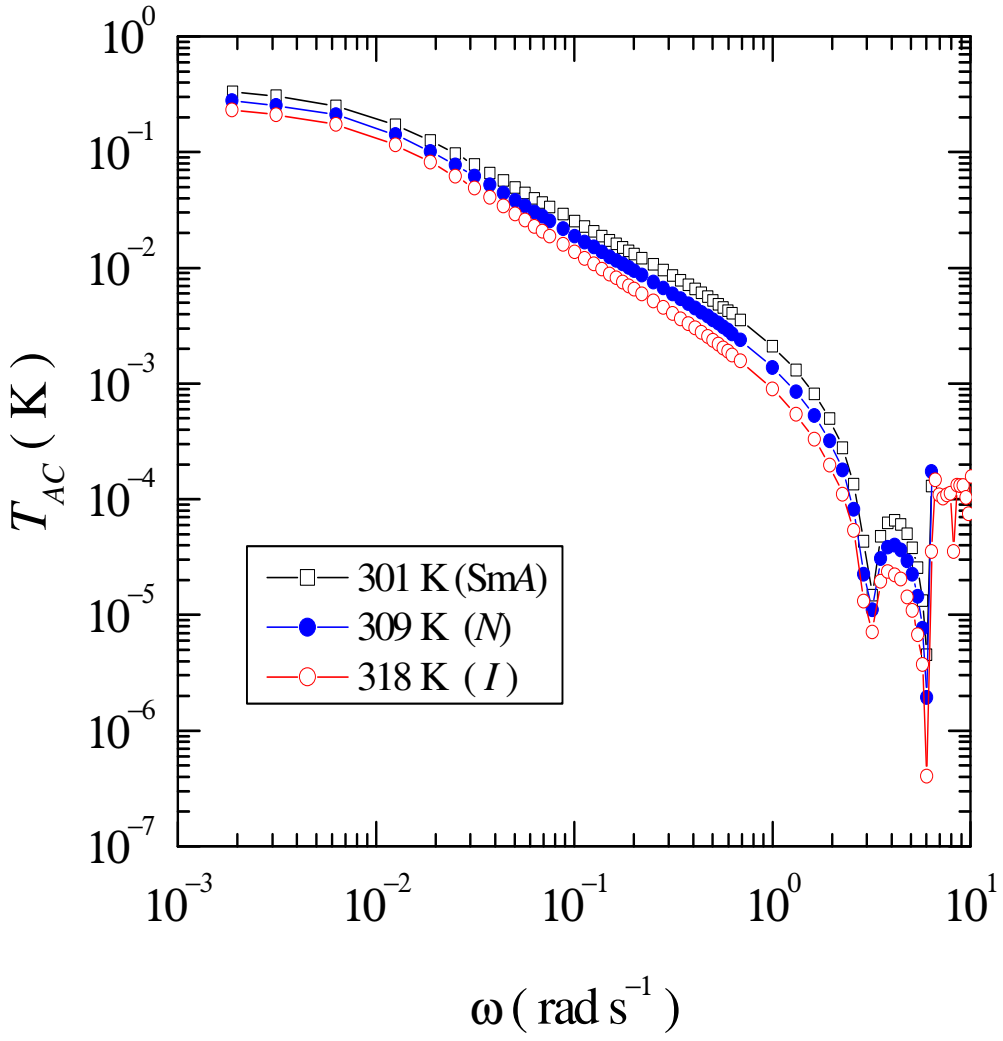


Figure 4.5: The dependence of the temperature oscillations amplitude T_{ac} from the modulation frequency at three different temperatures. As ω approaches zero, T_{ac} approaches T_{DC}.

In the frequency scan results shown in Figures 4.4 and 4.5, there is an interesting feature appearing for frequencies above 3.256 rad/s. This feature was never seen in the traditional AC calorimetry results. A very small plateau region is seen for all scans done at different temperatures. We can speculate at this time that this is a new feature related to the radio frequency heating method, and needs further considerations which are not part of the present work.

4.1.3 Temperature scan results.

After determining the operating frequency region for our calorimeter from the frequency scan results, we chose a modulation frequency of 15 mHz, or a period of 66.6 sec. for our temperature scan tests on 8CB + aerosil sample at 0.1 g/cm^3 silica density. The heat capacity of this sample was studied over a range of 17 K, from 301 to 318 K. A sharp first – order C_p peak at the Nematic – Isotropic transition, and a relatively rounded by distinct second – order C_p peak at the Nemanic – Smectic A transition are observed in the previous work [1] for the 0.1 g/cm^3 8CB + aerosil sample.

Traditional AC calorimetry by Joule's heating will show the temperature dependence of the heat capacity in the temperature scan data. Figure 4.6 shows typical temperature scan data taken by traditional AC calorimetry by Joule's heating in the 0.1 g/cm^3 8CB + aerosil sample. We show these data to compare it with our data of radio frequency heating calorimetry. The operating frequency of the calorimeter for these data

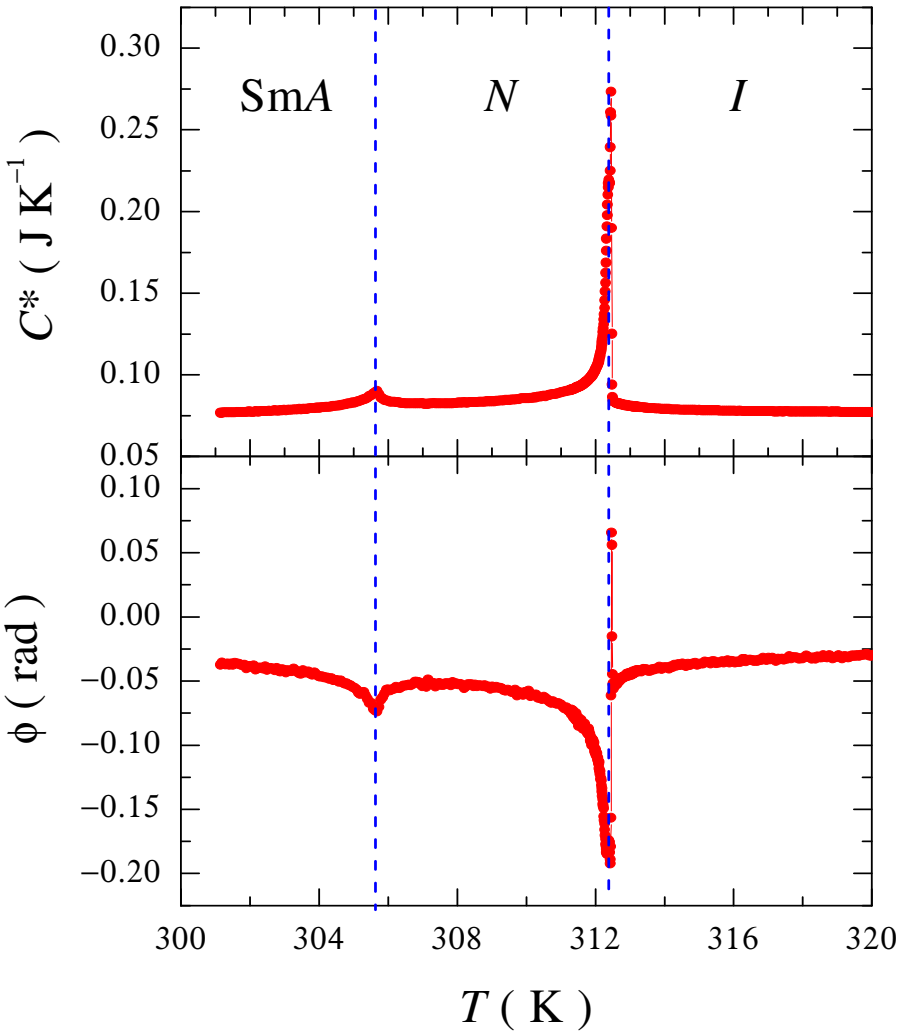


Figure 4.6: Typical temperature scan data taken with an AC calorimeter using a Joule heating method. The upper panel shows temperature dependence of the heat capacity from temperature, the lower panel shows temperature dependence of the phase shift.

was 15 mHz, the same frequency which we will choose to operate our test on the new AC calorimetry by radio frequency heating. Typical for traditional AC calorimetry is the very small dependence of the baseline heat capacity from temperature. This is clearly shown in the Figure 4.6. The nematic to isotropic phase transition occurs at 312.24 K, whereas the smectic-A to nematic phase transition occurs at 305.31 K.

In our new modulation calorimeter by radio frequency field heating, we will present the dependence of C^*/P_0, where P_0 is the amplitude of the applied power to the sample by radio frequency field heating from the temperature. Calibrating the power applied to the sample by radio frequency field heating will present challenges for another project.

Figure 4.7 shows a temperature scan data taken with our RF calorimeter. We have presented the dependence of C^*/P_0 from the temperature. Shown in that figure are the Nematic to Smectic-A phase transition at 305.3 K, and the Isotropic to Nematic phase transition at 312.2 K, the same transition temperatures as the traditional AC calorimetry data. The excess specific heat due to the SmA – N and N – I phase transitions and the C^*/P_0 nematic to isotropic wing behavior shown in Figure 4.7 for 8CB + aerosil sample is completely consistent with the previous results [2] on the same silica density, confirming the effect of quenched random disorder on phase structure and transitions. Calculations for C^*/P_0 were taken with a relative resolution of better than 0.06 %, up to 4 times better than previously reported, making our RF calorimeter an extremely high resolution technique for studying phase transitions.

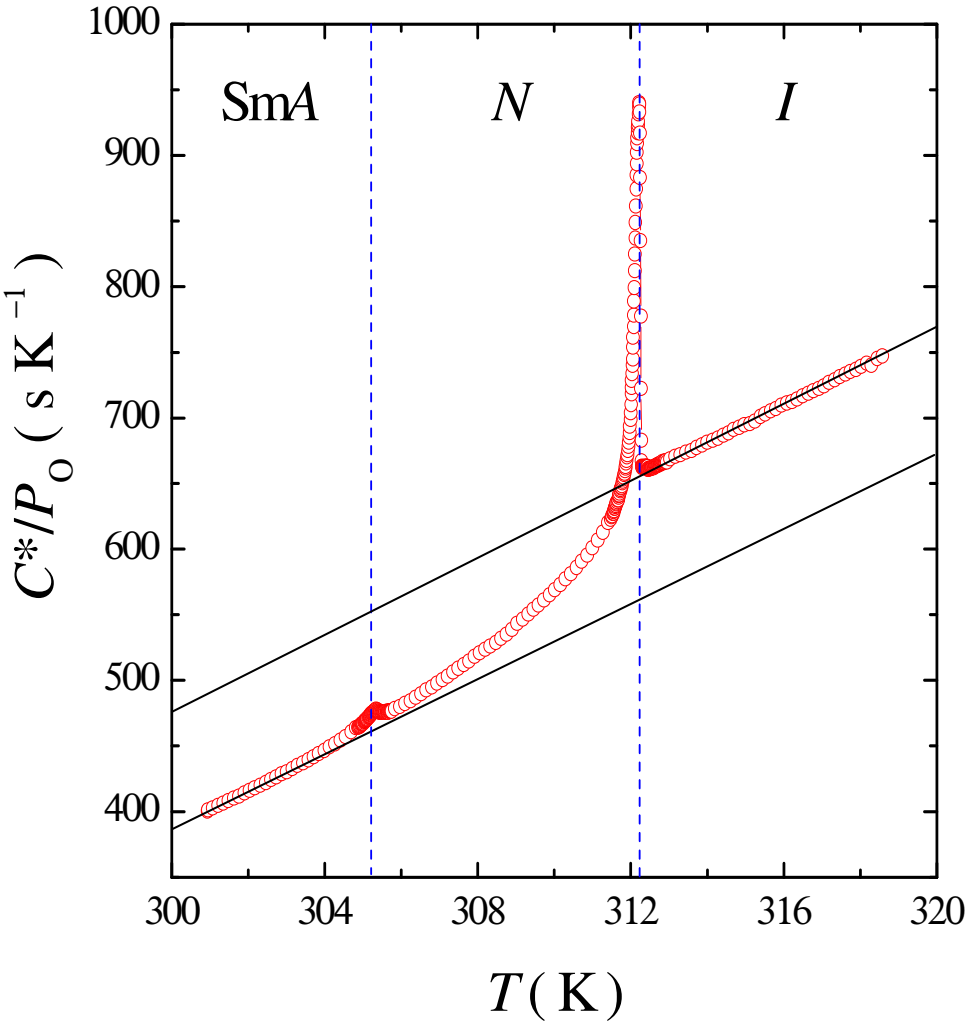


Figure 4.7: Temperature scan data performed on 8CB + aerosil sample at 0.1 g/cm^3 silica density. The continuous line represents the baseline of C^*/P_0, strongly temperature dependent.

In Figure 4.7 the continuous line represents the base line of the data. As can be seen clearly, there is a strong temperature dependence of this baseline from the temperature, and also a clear shift, or jump of the baseline in the Nematic to Isotropic phase transition region. As we showed in Figure 4.6, the traditional AC calorimetry by Joule's heating does not show a similar fact. The clear baseline shift and temperature dependence comes from the fact that we are presenting the dependence of C^*/P_0 in the AC calorimetry data by RF heating. The amplitude of the power P_0 applied to the sample, as shown from Eq. (2.28), depends on the permittivity $|\varepsilon|$ of the sample. The permittivity of a dielectric material is strongly temperature dependence, which shows clearly in our Figure 4.7 data. This gives us the opportunity to investigate both the heat capacity and permittivity dependence from the temperature of the sample. This is a new and exciting feature of our new AC calorimeter by radio frequency field heating.

The phase shift data taken at the same time for the 8CB + aerosil sample is presented in Figure 4.8, containing important information regarding the order of the phase transitions. Calculations for the phase shift data were taken with a relative resolution of better than 0.03 %, confirming again the very high resolution capabilities of our new calorimeter technique. A very sharp peak in the phase shift can be observed at the first order nematic to isotropic phase transition, coinciding with the first order phase transition in the heat capacity data. This is the region of coexistence of two phases for a first order phase transition in the sample. Coinciding with the second order smectic-A to nematic phase transition in the heat capacity data, we can see a small decrease in the phase shift due to the absence of the latent heat and a correlation length diverging to macroscopic length scales.

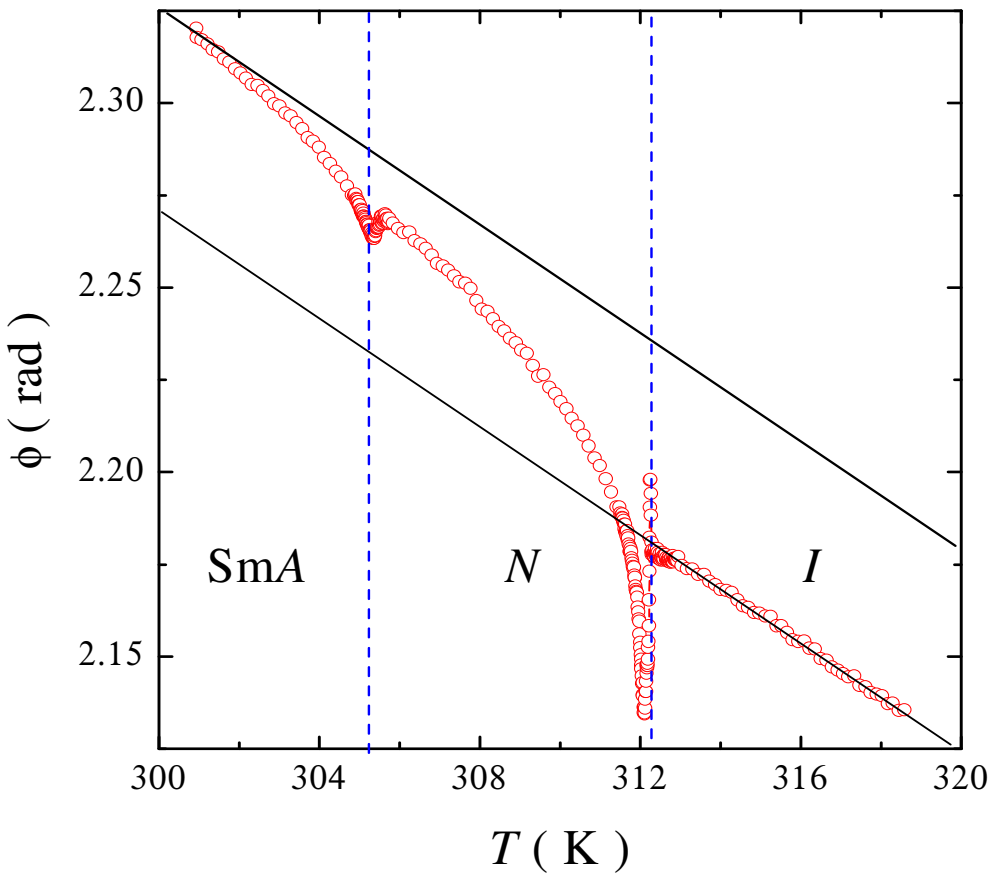


Figure 4.8: The phase shift temperature dependence data for 8CB + aerosil sample at 0.1 g/cm^3 silica density. The continuous line represents the baseline of the phase shift, strongly temperature dependent.

In Figure 4.8 the continuous line represents the base line of the data. As it is seen clearly, again there is a strong temperature dependence of this baseline from the temperature, and also a clear shift, or jump of the baseline in the Nematic to Isotropic phase transition region. Eq. (3.13) clearly indicates that the phase shift between the applied power and temperature oscillations should not depend on the power applied to the sample. Figure 4.8 clearly shows a temperature, thus power dependence of the phase shift data. The investigation of this new feature of the phase shift data of our new RF heating calorimetry technique would present the basics for future projects.

4.2 Radio Frequency calorimetry results on 8CB bulk and 8CB + aerosil dispersions.

The study of the effect of quenched random disorder (QRD) on physical systems has an important interest in the physics of soft condensed matter. The general understanding is that the physics of quenched random disorder in liquid crystals is contained by a random – field approach [3]. The random perturbations are introduced via the embedding of a random (gel-like) solid structure into the phase ordering material. A very detailed study of the strength of the random disordering field as a function of the concentration of such solid is presented in [1], and shows that the amount of the orientational order that exists in the dispersions decreases linearly with increasing silica density. The effect of aerosil – induced random disorder on LC phase behavior can be categorized into two regimes: the first one is characterized by rigid aerogel behavior, where the first and second order phase transitions suffer from quenched elastic – strain

smearing effects with a radius of curvature of the elastic distortion taken as much larger than the mean void size. Examples of such effects are presented for 8CB in References [4-8].

The second regime that emerges is the low – density liquid crystal + aerosil dispersions, which can be seen as the soft – gel regime. In this case, the second order nematic to smectic-A phase transition heat capacity peak remains sharp with its critical behavior evolving toward 3D XY universality in accordance with the Harris criterion [9], while rapidly shifting to lower temperatures. Also, the first – order nematic to isotropic phase transition remains sharper in low – density aerosil mixtures [10] than in any aerogel samples of high – density aerosil samples. Plus, there exist two peak features at the first order nematic to isotropic phase transition [1], a very sharp and narrow spike followed at lower temperatures by a more rounded peak.

Recent deuteron nuclear magnetic resonance studies [11], have shown that as a function of silica density up to low density stiff-gel dispersions, cooling the samples in the presence of a strong magnetic field leads to a rearrangement of silica links such that disorder – induced elastic strains are greatly annealed in the vicinity of nematic to isotropic phase transition. But this is not so much in the smectic phase, due to the shorter domain – limited smectic correlation length and the longer magnetic coherence length.

In this work we present high resolution radio frequency heating calorimetric study of the liquid crystal phase transitions for octylcyanobiphenyl (8CB) bulk, and for several dispersions of 7 nm diameter silica spheres (aerosil) in 8CB as a function of silica density ρ_S. Using radio frequency heating calorimetry, we put the liquid crystal under the influence of the electric field. In soft gel regime, where the amount of disorder

introduced by the silica particles is minimal, the N - I and SmA – N phase transitions retain their sharpness, but for the stiff gel regime, where the disorder is large and fully quenched, the phase transitions are smeared and SmA – N phase transition is completely suppressed.

Several radio frequency heating calorimetry runs were carried out for bulk 8CB and for several densities of aerosil particles in 8CB: 0.03, 0.1 and 0.2 g/cm^3. Figure 4.9 shows as a typical example of the C^*/P_0 variation over and extended temperature range for 8CB bulk sample. Data was obtained for this sample at the angular frequency of $\omega = 0.0942 \frac{rad}{s}$, or f = 0.015 Hz. Sharp Nematic to Isotropic and a step – Smectic-A to Nematic phase transitions are observed for the bulk at respectively 313.0 K and 306.1 K, as shown clearly in Figure 4.9. The simultaneously – taken phase shift data for the bulk 8CB sample is presented in Figure 4.10, containing important information regarding the order of the phase transitions.

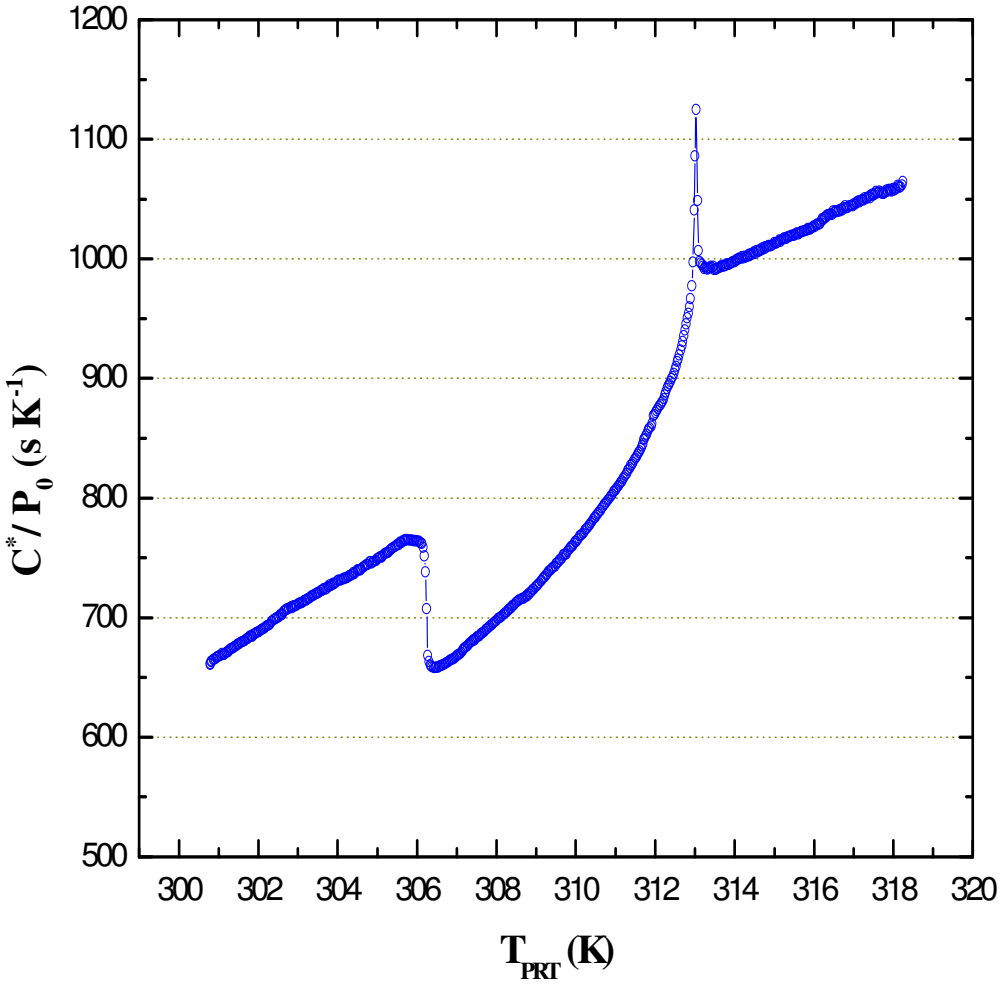


Figure 4.9: Temperature scan data performed on 8CB bulk sample. Nematic to isotropic and a Smectic-A to Nematic phase transitions are observed for the bulk at 313.0 and 306.1 K, respectively.

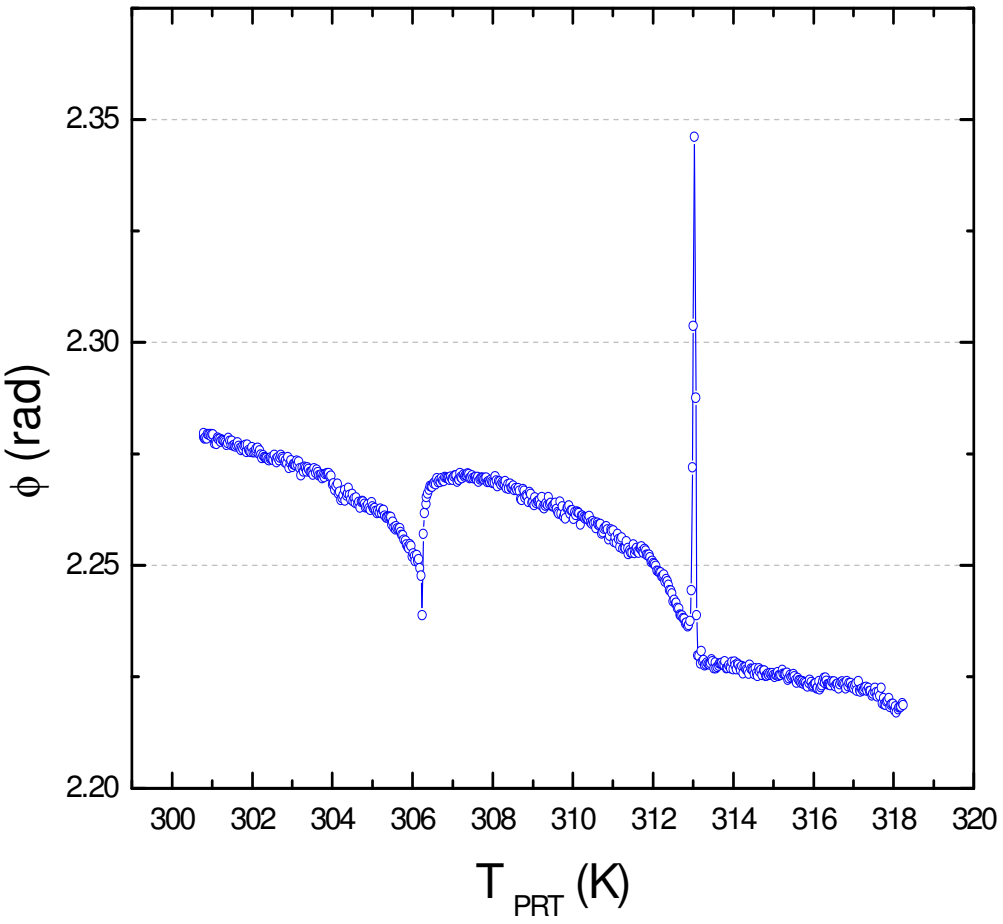


Figure 4.10: The phase shift temperature dependence data for bulk 8CB sample. The signatures of Smectic-A to Nematic and Nematic to Isotropic phase transitions are clearly seen.

A very sharp peak in the phase shift can be observed at the first order nematic to isotropic phase transition, coinciding with the first order phase transition in the heat capacity data. This is the region of coexistence of two phases for a first order phase transition in the sample. Coinciding with the second order smectic-A to nematic phase transition in the heat capacity data, we can see a decrease in the phase shift due to the absence of the latent heat and a correlation length diverging to macroscopic length scales. The step-like character of this transition is clearly indicated by the phase shift data too, and comes from the dominance of the permittivity and the loss factor parameters in the C^*/P_0 data. This step-like character of the smectic-A to nematic phase transition represents the dielectric coupling between the nematic and smectic-A phases. This is understandable, going from nematic to smectic-A phase, the molecules of the liquid crystal gain more orientational and positional order, the domains in the system align themselves, and accordingly there will be a jump in the permittivity of the material, which is clearly showing up in our data.

Several radio frequency heating calorimetry runs were carried out for bulk 8CB and for several densities of aerosil particles in 8CB: 0.03, 0.1 and 0.2 g/cm^3. In Figure 4.11 we present different temperature scan data taken for the above samples and for bulk 8CB. An expanded view of the temperature scan data about the nematic to isotropic and smectic-A to nematic phase transitions is presented in Figures 4.12 and 4.13 respectively, as a function of temperature. Temperature shifts in the C^*/P_0 peaks and suppression of the nematic to isotropic phase transition are seen in Figure 4.14 as the density of silica is increased. For the smectic-A to nematic phase transition, we have subtracted the nematic background wing. Figure 4.13 clearly shows a change from step-like, occurring for bulk

8CB, to a smooth second order transition for 0.03 and 0.1 g/cm^3 samples, and then total disappearance of the transition for 0.2 g/cm^3 sample.

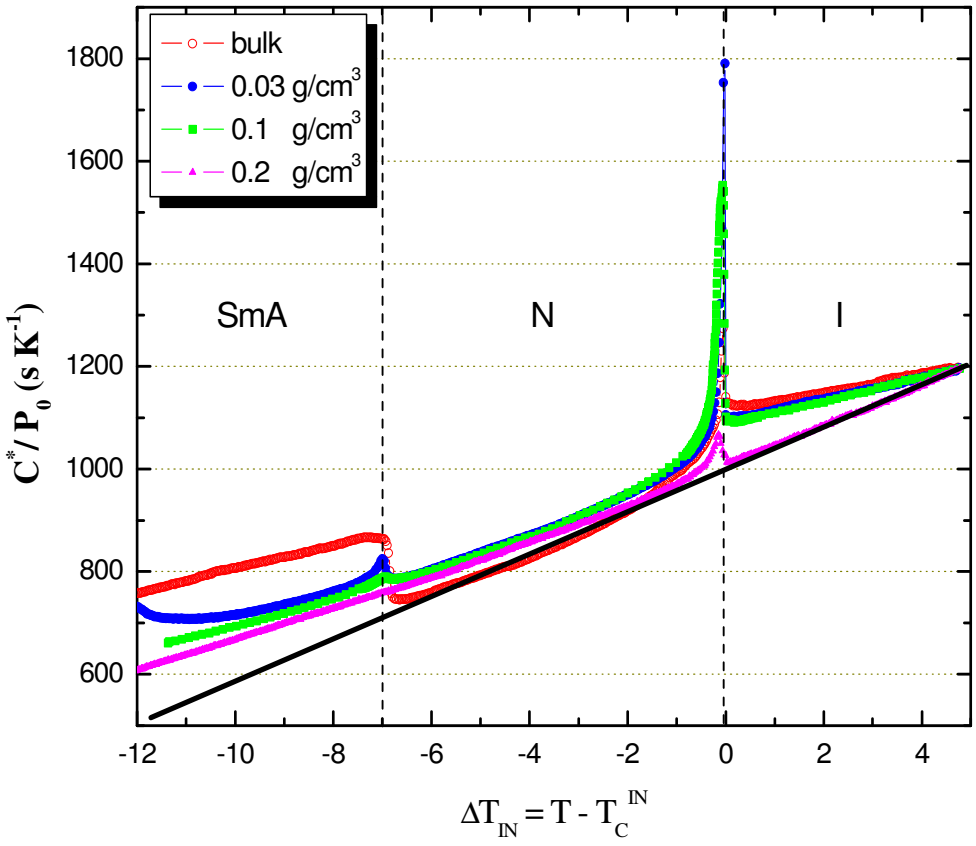


Figure 4.11: C^*/P_0 as a function of temperature about T_{IN} for 8CB bulk and 8CB + aerosil samples at different silica densities, 0.03, 0.1, and 0.2 g/cm^3 of liquid crystal. Smectic-A, Nematic and Isotropic regions are clearly indicated in the figure. The baseline of these data is scaled to a point in the isotropic region with $\Delta T_{IN} = 4.8K$.

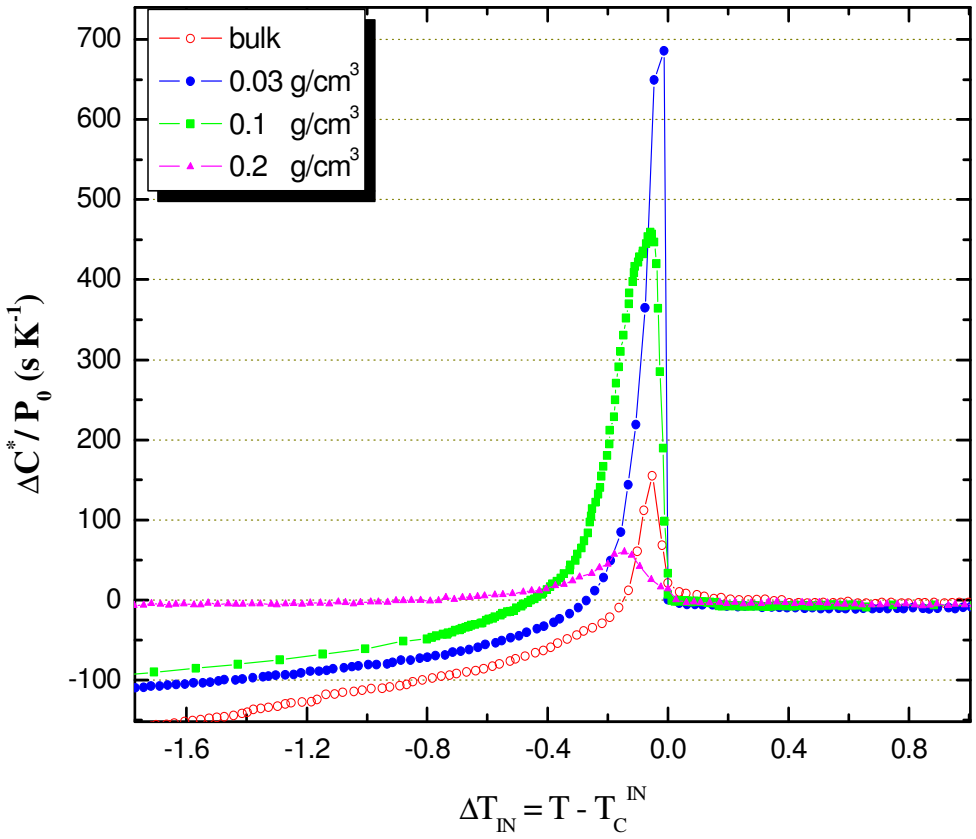


Figure 4.12: Expanded view of the temperature scan data about the nematic to isotropic phase transition as a function of temperature. We have subtracted isotropic background wing. See figure insets for definition of symbols. The evolution of the first order isotropic to nematic phase transition of C^*/P_0 peak is clearly indicated from the picture starting from the bulk, getting bigger for 0.03 g/cm^3 sample, getting suppressed for the 0.1 g/cm^3, and more suppressed for 0.2 g/cm^3 sample.

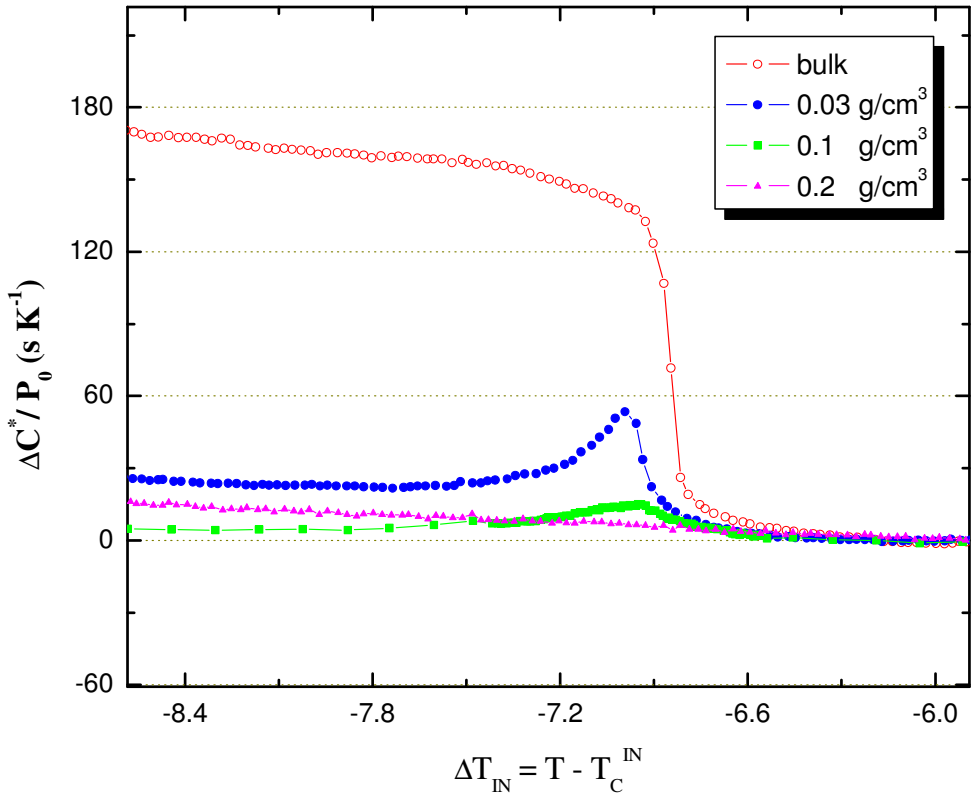


Figure 4.13: Expanded view of the temperature scan data about the nematic to isotropic phase transition as a function of temperature. We have subtracted the nematic background wing. See figure insets for definition of symbols. The evolution of the second order smectic-A to nematic phase transition is clearly indicated from the picture starting from the bulk, as a step-like character, getting smaller and forming a peak character for 0.03 g/cm^3 sample, getting suppressed for the 0.1 g/cm^3, and smeared out for 0.2 g/cm^3 sample.

For all the data shown in Figures 4.11 – 4.13 the baseline for the C^*/P_0 is scaled to a point well inside the isotropic phase, with $\Delta T_{IN} = 4.8K$. We chose this point because the isotropic phase is well defined and the data should match for the bulk and the other 8CB + aerosil samples at different silica densities. Comparing our data with the one done previously [1], one can immediately see the differences. For the traditional AC calorimetry data [1], the size and shape of the pre - transitional wings $\Delta C_p(NI)$ are independent of the silica density, in spite of significant shifts in the T_{NI} transition temperatures. Our radio frequency heating modulation calorimetry data in Figures 4.11 and 4.12 clearly shows a mismatch of the C^*/P_0 pre - transitional wings. Again would like to point out that this feature comes from the fact that the amplitude of the power applied to the sample given by Eq. (2.28), depends heavily from the permittivity $|\varepsilon|$ and the loss factor $\tan\delta = \frac{\varepsilon_2}{\varepsilon_1}$ of the material, where ε_1 and ε_2 are the real and complex parts of the perimittivity of the samples respectively. At different temperatures for different samples, these factors are quite different, as clearly indicated from the data. This mismatch in the C^*/P_0 pre - transitional wings confirms our simultaneous probing of the permittivity and the heat capacity of the samples, not doable with the traditional AC calorimetry technique.

A very dramatic domination of the dependence of the data from the permittivity $|\varepsilon|$ and the loss factor $\tan\delta = \frac{\varepsilon_2}{\varepsilon_1}$, can clearly be seen in the smectic-A to nematic transitions shown in Figure 4.13. The step-like character of this transition shown in Figure 4.13 for the bulk sample comes from the dominance of the permittivity $|\varepsilon|$ and the

loss factor $\tan\delta = \frac{\varepsilon_2}{\varepsilon_1}$ parameters in the C*/P_0 data. This dominance is then suppressed in the 8CB + aerosil at 0.03 and 0.1 g/cm^3 silica densities, where a "peak" character of the smectic A to nematic phase transition is revealed. No feature is revealed for the smectic A to nematic phase transition for 0.2 g/cm^3 silica density sample, in agreement with the previous traditional AC calorimetry data [1]. The coupling between the smectic-A and nematic phases decreases with increasing silica density. The evolution of the smectic-A to nematic phase transition also indicates that the nematic susceptibility is being suppressed with increasing silica density. The effect of quenched random disorder is clearly showing in the data: the shift in transition temperatures, the broadening of the transition region, the suppression of the peak with increasing silica density, and even the complete disappearance of the peak for the 0.2 g/cm^3 sample.

Comparing our C*/P_0 radio frequency heating calorimetry data for nematic to isotropic phase transition shown in Figure 4.12, with the one in reference [1], one could notice the differences. A large sharp peak for the nematic to isotropic phase transition can be seen in the C* traditional AC calorimetry data [1]. The height of this peak then decreases with increasing silica density in the 8CB + aerosil sample. In our radio frequency heating calorimetry data for the nematic to isotropic phase transition shown in Figure 4.12, the first order nematic to isotropic phase transition peak is clearly suppressed because of the domination of the permittivity $|\varepsilon|$ and the loss factor $\tan\delta = \frac{\varepsilon_2}{\varepsilon_1}$ parameters. A sharp peak is then observed for the 8CB + aerosil sample at 0.03 g/cm^3 silica density, followed by a suppressed nematic to isotropic peak for the other 8CB + aerosil samples at 0.1 and 0.2 g/cm^3 silica densities. The shift in transition temperatures,

the broadening of two phase coexistance region, and the suppression of the peak with increasing silica density indicate the clear quenched random disorder effect in the system. This is in total agreement with the previous AC calorimetry data from reference [1]. What we don't see in this data is the doble feature of the N-I peak observed in the previous work done by traditional AC calorimetry technique. This feature is being suppressed by the permittivity response of our data.

References:

[1] G. S. Iannacchione, C. W. Garland, J. T. Mang, and T. P. Rieker, Phys. Rev. E 58, 5966 (1998).

[2] G. S. Iannacchione, S. Park, C. W. Garland, R. J. Birgeneau, R. L. Leheny, Phys. Rev. E 67, 011709 (2003)

[3] A. Pelissetto and E. Vicari, Phys. Rev. B 62, 6393 (2000).

[4] T. Bellini, N. A. Clark, C. D. Muzny, L. Wu, C. W. Garland, D. W. Schaefer, and B. J. Oliver, Phys. Rev. Lett. 69, 788 (1992).

[5] X.-I. Wu, W. I. Goldburg, M. X. Liu, and J. Z. Xue, Phys. Rev. Lett. 69, 470 (1992)

[6] N. A. Clark, Phys. Rev. Lett. 71, 3505 (1993)

[7] G. S. Iannacchione, S. Qian, D. Finotello, and F. Aliev, Phys. Rev. E 56, 554 (1997)

[8] L. Wu, B. Zhou, C. W. Garland, T. Bellini, and D. W. Schaefer, Phys. Rev. E 51, 2157 (1995).

[9] A. B. Harris, J. Phys. C 7, 1671 (1974); J. T. Chayes, L. Chayes, D. S. Fisher, and T. Spencer, Phys. Rev. Let 57, 2999 (1986).

[10] B. Zhou, G. S. Iannacchione, C. W. Garland, and T. Bellini, Phys. Rev. E 55, 2962 (1997).

[11] T. Jin and D. Finotello, Phys. Rev. E 69, 041704 (2004).

CHAPTER 5

ISOTROPIC TO NEMATIC TRANSITION OF AEROSIL – DISORDERED LIQUID CRYSTALS

5.1 INTRODUCTION

The study of ordering transitions in systems within disordered environments has provided new insights into the physics of phase transitions [1, 2]. Liquid crystals (LCs) incorporating low-density gels formed by silica aerosil dispersions (LC+A) have been shown to be good experimental models to investigate both the isotropic (I) to nematic (N) and the nematic to smectic-A (SmA) phase transitions in the presence of the static (quenched) positional and orientational randomness provided by surface coupling at the silica-LC interfaces [3-6]. While most of the observed behavior at the N-SmA phase transition in LC+A - scaling laws, structure factors, pseudo-critical exponents - have been understood on the basis of the current models treating the statistical physics of random-field systems [4-6], the I-N transition in LC+A is more difficult to interpret. This is in part due to the first-order nature of the bulk transition where random disorder introduces a distribution of transition temperature shifts that broadens two-phase coexistence [7], as well as to the quadrupolar nature of the nematic order parameter [8]. A particular

challenge is the observed doubling of the LC+A calorimetric peak at the I-N transition in an intermediate range of silica concentrations [8-10]. This feature is interesting because it has no analogy in the LC bulk behavior, and thus its investigation provides a unique route to access the basic physics of discontinuous transitions in artificially disordered systems.

The three-dimensional nematic orientational order, established from the *I* state through a weakly first-order phase transition, is described by a symmetric and traceless 2nd rank tensor Q_{ij}. For uniaxial systems it may be approximated on short length scales by a scalar order parameter (S) measuring the magnitude of order and on longer length scales by the director $\hat{n}$ describing the spatial variation of the orientation axis [11]. The bulk transition exhibits significant pretransitional fluctuations and is very close to a tricritical point [12]. Both aspects are nontrivially affected by random disorder.

In this work we present an experimental study of LC+A samples made with a low birefringence material, 4' – transbutyl – 4 – cyano – 4 – heptyl - bicyclohexane (CCN47), in which the biphenyls group is replaced by saturated hydrocarbon analogs, yielding a birefringence (Δn) about 1/10 that of cyanobiphenyls. The use of such a material enables a better optical characterization both through the study of I(q) in the single scattering regime - where the nematic (director) correlation length ξ_N can be extracted - and through the measurement of the turbidity $\tau(T)$ in the Rayleigh-Gans regime, in which the turbidity is a simple function of the relevant properties of the local nematic, namely $\tau(T) \propto \phi_N(T)\Delta n^2(T)\xi_N(T)$ [13] where ϕ_N is the nematic volume fraction. We have thus undertaken a combined T-dependent optical and calorimetric investigation of CCN47 LC+A samples through the I-N transition over a range of silica densities

displaying the double I-N transition peak. This work offers compelling evidence that the I-N transition with weak quenched random disorder proceeds via a two- step process, in which random-dilution is followed by random-field interactions on cooling from the isotropic phase, a previously unrecognized phenomena.

5.2 ISOTROPIC TO NEMATIC TRANSITION OF CCN47 + AEROSIL DISORDERED SYSTEMS.

The CCN47 liquid crystal features a wide range nematic (N) phase between isotropic (I) and smectic-A (SmA) phases. According to our data, T_{IN} = 330.21 K and T_{NA} = 301.63 K, is in agreement with previous investigations [14]. Measurements of the bulk birefringence Δn_B by birefringence compensation at the microscope can be described by $\Delta n_B = \Delta n^{**} + A\left(T^{**} - T\right)^{\beta}$, where $\Delta n^{**} = 0.0085$, $A = 0.0096$, $T^{**} - T_{IN} \approx 0.14$ K, and $\beta \cong 0.28$, consistent with typical values for nematics [15]. Characterization of the I-N transition by calorimetry reveals a weak first-order character, with a latent heat $\Delta H \approx 1.55 Jg^{-1}$ and a total transition enthalpy (including the pretransitional contribution δH) $\Delta H_T = \delta H + \Delta H \approx 5.65 Jg^{-1}$.

We performed a systematic investigation of CCN47+aerosil samples prepared by incorporating dried hydrophilic silica nanoparticles forming a mass-fractal gel (Degussa, Aerosil type 300, 7 nm diameter) at various silica densities (0.050, 0.075, 0.100, and 0.200 g SiO_2 per cm^3 LC; hereafter we drop the units) via evaporation of an intermediate solvent mixture [9]. The same samples have been used for AC and Non-Adiabatic

Scanning (NAS) calorimetry, optical turbidity, Integrated Low Angle Light Scattering (ILALS), as well as for optical microscopy observation. The AC and NAS calorimetry were performed in a multi-mode calorimeter on the same sample + cell arrangement [16].

Since heat capacity measures the entire spectrum of energy fluctuations ranging from short range molecular interactions to fluctuations in long-range order (S and $\hat{n}$ for nematics), direct comparisons with other experiments are sometimes difficult. In particular, optical studies naturally probe the nematic order of sub – micron size and larger. For a quantitative comparison, the contribution to C_P related to optical length scales, i.e. in a bulk-like manner, is required. Typically for studies of first-order transitions, C_P (AC) is truncated through the two-phase coexistence region, then subtracted from C_P (NAS), which is then integrated to determine the transition latent heat ΔH [6, 9]. The two techniques are completely consistent outside the two-phase coexistence region. However, disorder can seriously alter the dynamics and T-width of the two-phase conversion region allowing C_P (AC) to sense a significant fraction of the transition latent heat. By foregoing the usual truncation of C_P (AC), we extract $\Delta C_B = C_P(NAS) - C_P(AC)$ that can be associated with domain conversions slower than the AC cycle. Since our AC cycle is already quite slow (15 mHz), these domains should be large. Its integration gives the enthalpy $\Delta H_B(T, \rho_S) = \int \Delta C_B dT$, which we propose to be directly proportional to the fractional conversion of I into N, ϕ_N, as optically measured through the two-phase coexistence region.

The turbidity $\tau(T)$ was measured by spatial filtering in the forward direction the light transmitted through a LC+A sample [3]. The ILALS was measured by collecting all the depolarized light scattered within a cone limited by the scattering vector q into an

integrating sphere. Selecting depolarized light suppresses the largest part of transmitted light and of the light scattered by the silica structure. This simple technique, though less precise than the direct measurement of I(q), still permits a reliable measurement of the director correlation length ξ_N in the single scattering regime [17].

The $\Delta C_B(T)$ variation for ρ_S = 0.075 is shown in Figure 5.1 as a function of the temperature shift $\Delta T = T - T_{IN}(\rho_S)$, where $T_{IN}(\rho_S)$ is the temperature of the first appearance of nematic in LC+A samples. The double calorimetric feature is thus evident here as it was for other LCs [8, 9]. Figure 5.1 also shows for the same sample, in a cell $48\mu m$ thick, $\tau(T)$ (left axis) together with the bulk behavior of $\Delta n^2(T)$ (right axis). The match between $\tau(T)$ and $\Delta n^2(T)$ extends over a large T interval at a lower temperature (inset Figure 5.1b), indicating that, outside the transition region, ξ_N is constant, as previously reported [13]. An analogous matching is found over the same T region, between the ΔC_P measured for the bulk and the LC+A CCN47 samples, which is completely consistent with data reported in Ref. [9]. This confirms that outside the immediate T region of the transition, the T dependence of LC+A systems is entirely expressed by the variation of a bulk-like local order parameter S, with no director or silica restructuring. In the I+N coexistence region, because of the distributed local transition temperatures, the determination of the Δn^2 for the N-fraction, with which to interpret $\tau(T)$, is more difficult. In what follows, we have adopted the simplest view and assume that the value of Δn^2 for the nematic fraction in the two-phase coexistence region is given by a linear extrapolation of the bulk Δn^2 (dashed line in Figure 5.1b).

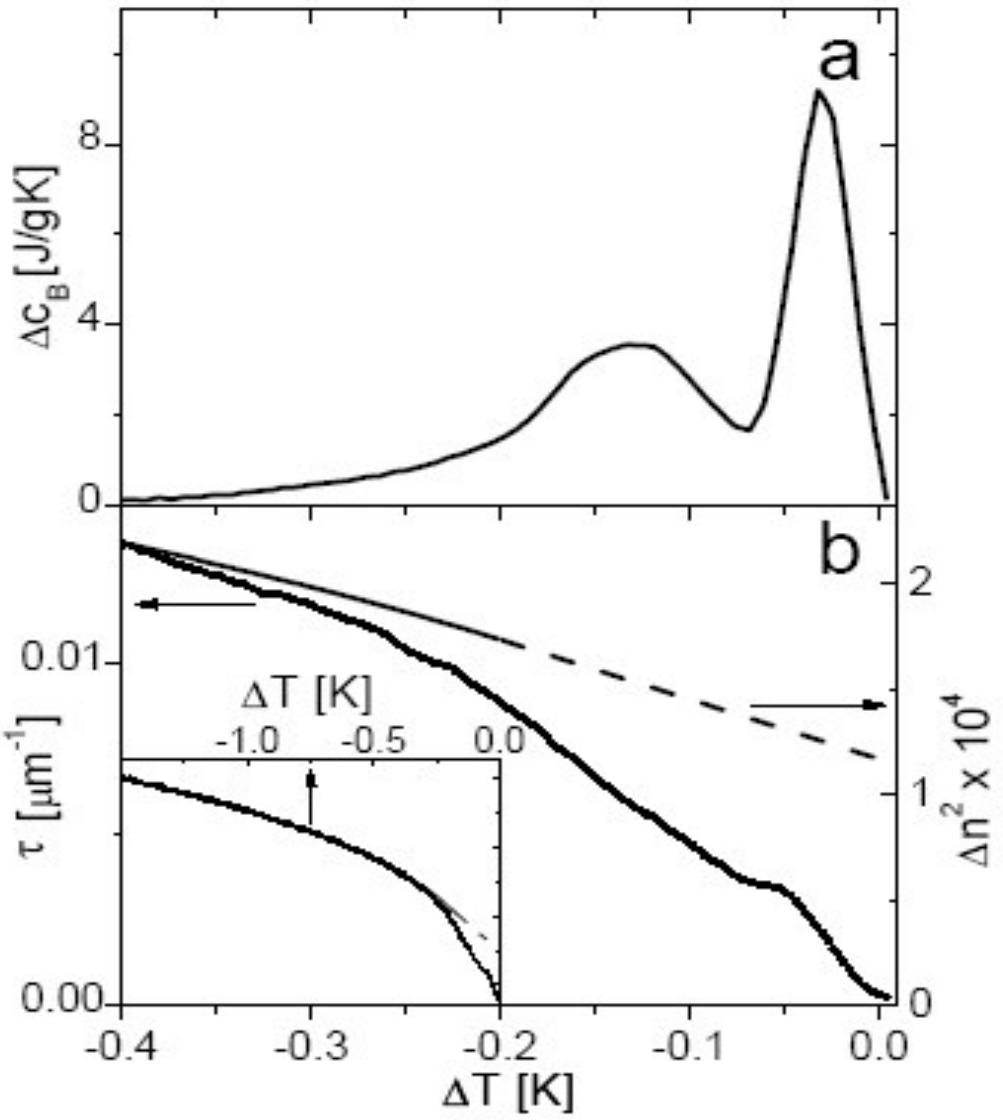


Figure 5.1: Excess specific heat ΔC_B (a) and turbidity τ [(b) left axis] measured as a function of the temperature shift ΔT within the two-phase coexistence region for the LC + A ρ_S =0.075 sample. Also shown, the bulk birefringence squared Δn^2 [(b) right axis]: measured values (solid line) and linear extrapolation (dashed line). Inset: τ and Δn^2 over a wide ΔT range.

This is supported by NMR data on LC+A indicating that in the coexistence region the local S has a weaker T dependence than bulk [18]. The ξ_N (T) extracted from ILALS data following Ref. [13], is shown in Figure 5.2. The data was obtained in a $48\mu m$ thick sample in which the fraction of the collected light resulting from a single scattering process is, in the plotted range, always larger than 71%.

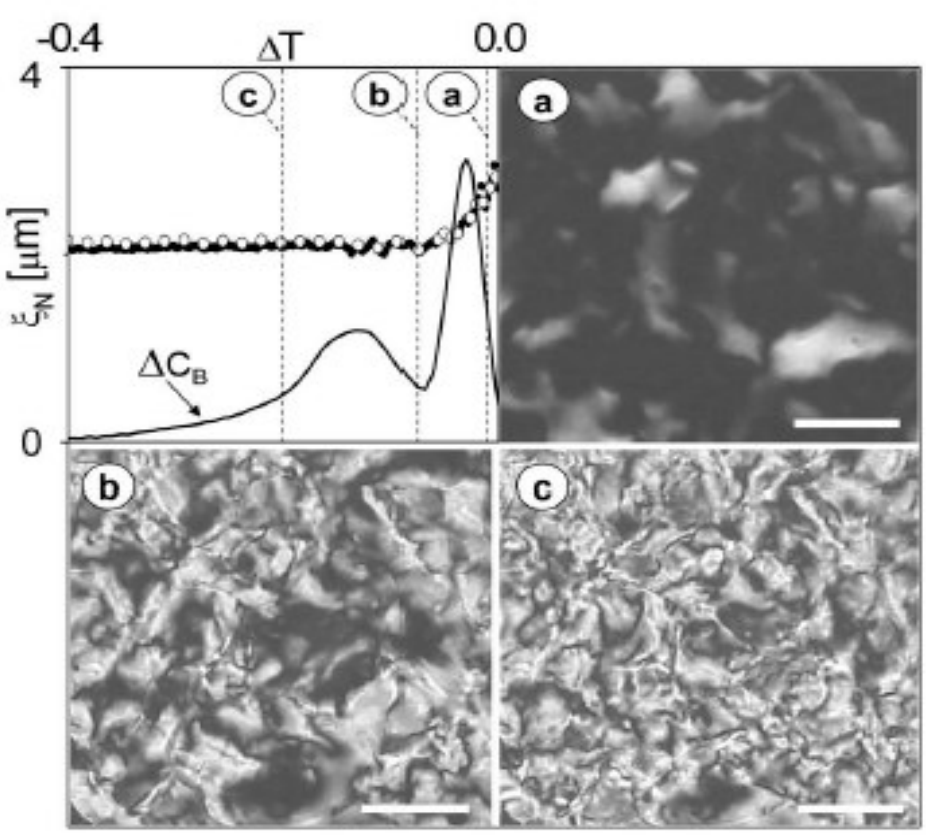


Figure 5.2: Upper Left Panel: Nematic correlation length ξ_N raw data (solid dot) and double scattering corrected (open dot) and ΔC_B (arbitrary scale) for the ρ_S = 0.075 sample measured as a function of ΔT. Panels (a), (b), (c): Optical cross-polarized microscope pictures taken at a ΔT of -0.033 (a), -0.07 (b), and -0.2 K (c) indicated by the vertical dotted lines in the upper left panel. The bar corresponds to $10\mu m$.

As shown in the Figure 5.2, first-order correction (double scattering - open dots) has a negligible effect on ξ_N. From the optically measured τ, Δn^2 and ξ_N, we extract $\phi_N(T)$ shown in Figure 5.3.

As Fig. 5.3 demonstrates, both calorimetric and optical measurements indicate that the double peak must be entirely due to a two-step transformation of the LC from the isotropic to the nematic state. The results from the calorimetric and optical techniques display remarkable agreement. The relative size of the two processes is plotted for samples having different ρ_S in the inset of Figure 5.3. Apart from the 0.050 sample, which may suffer from the separation of bulk LC from the LC+A during the thin cell preparation, the agreement is very good. To interpret the double transition process, a key observation is the smoothness and the limited decrement of ξ_N data, which unambiguously demonstrate that the transition doubling cannot be attributed to a bimodal distributed porosity in the silica gel. However, the decrease of ξ_N as T decreases, which has been confirmed by direct optical microscopy observations shown in Figure 5.2, offers an important clue for interpreting the observed behavior. Figures 5.2a-c show a sample in cross-polarized microscopy at three different temperatures corresponding to the first appearance of the N phase, to a point in between the C_P peaks, and to a point near the low T end of the two-phase coexistence region that remains unchanged to lower T (see lower right panel of Figure 5.2). The nematic first appears as isolated regions with approximately uniform director alignment.

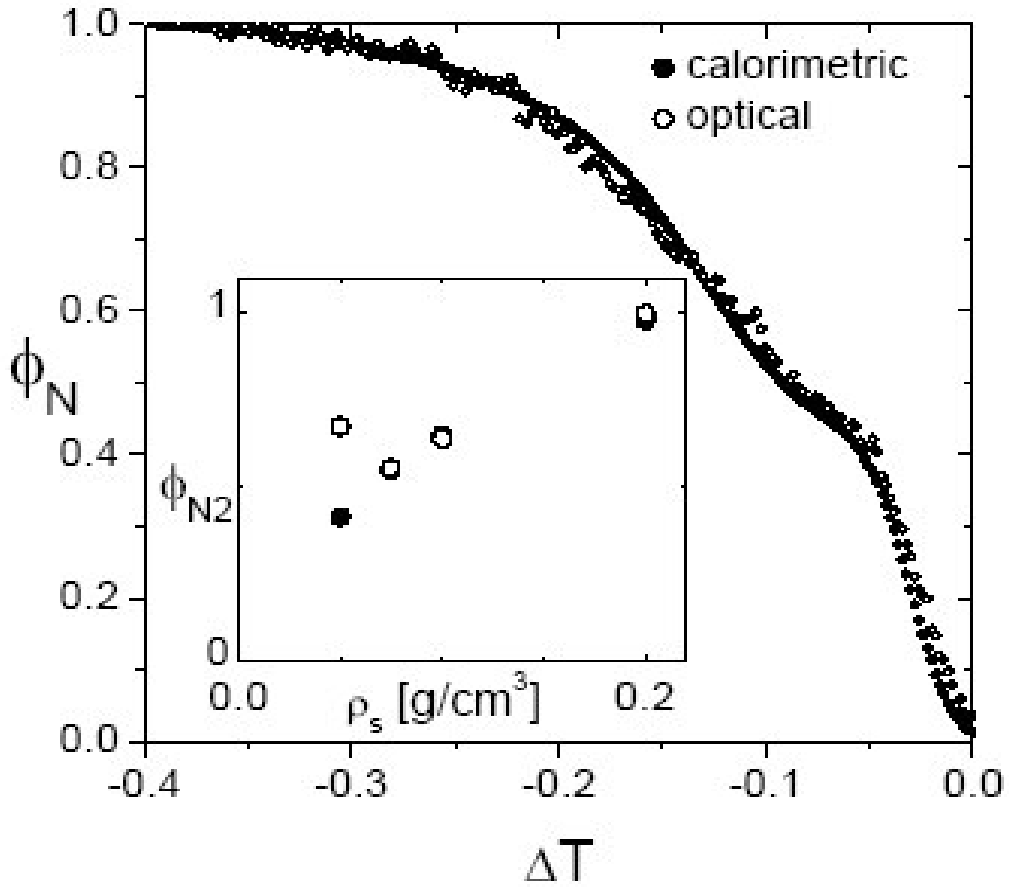


Figure 5.3: Nematic volume fraction ϕ_N obtained from the integral of ΔC_B (0.075) (solid dots) and deduced from optical measurements $\phi_N \propto \frac{\tau}{\left(\Delta n^2 \xi_N\right)}$ (open dots) as a function of ΔT through the two-phase coexistence region. Inset: N volume fraction ϕ_{N2} converted through the low temperature C_P peak, as a function of ρ_S.

As T decreases through the region of the higher-T C_P peak, the domains grow in number, distort and exhibit recognizable defect lines, an indication of an increased coupling of the director with the silica surfaces. As this distortion proceeds, the transition slows and, for a narrow interval between the C_P peaks, no new nematic domains appear. As T decreases further, the nematic growth manifests itself in a different way. New volumes undergo the I-N transition, but no changes in the director are observed: domains appear with their "final" low T distorted structure. This is consistent with an essentially constant correlation length thereafter. Thus, the combination of microscope observations and the ξ_N data suggest that the two processes differ in the strength of the coupling with the silica matrix.

We argue that this observation leads to an explanation of the nature of the double C_P peak. The incorporation of solids in a spontaneously ordering media has different effects on the phase behavior and, in particular, on the shift and distribution of transition temperatures. The simplest expected - and, in some cases, observed [19] - effect comes from dilution, weakening of the mean molecular field. At the mean field level, this effect downshifts T_{IN} by the total reduction of the molecular coordination, provided by the contact with the solid-LC interface area. The disorder broadening of the transition can be most simply modeled by a distribution in ΔT_{IN} within the two-phase coexistence region linear in the local disorder (silica) density (dashed line in Figure 5.4b). If this was the only mechanism, the amount of material undergoing the I-N transformation in a given interval within the two-phase range would simply be an image of the disorder density distribution function, possibly averaged by a suitable nucleation length, in analogy to that described in other phase transitions [20].

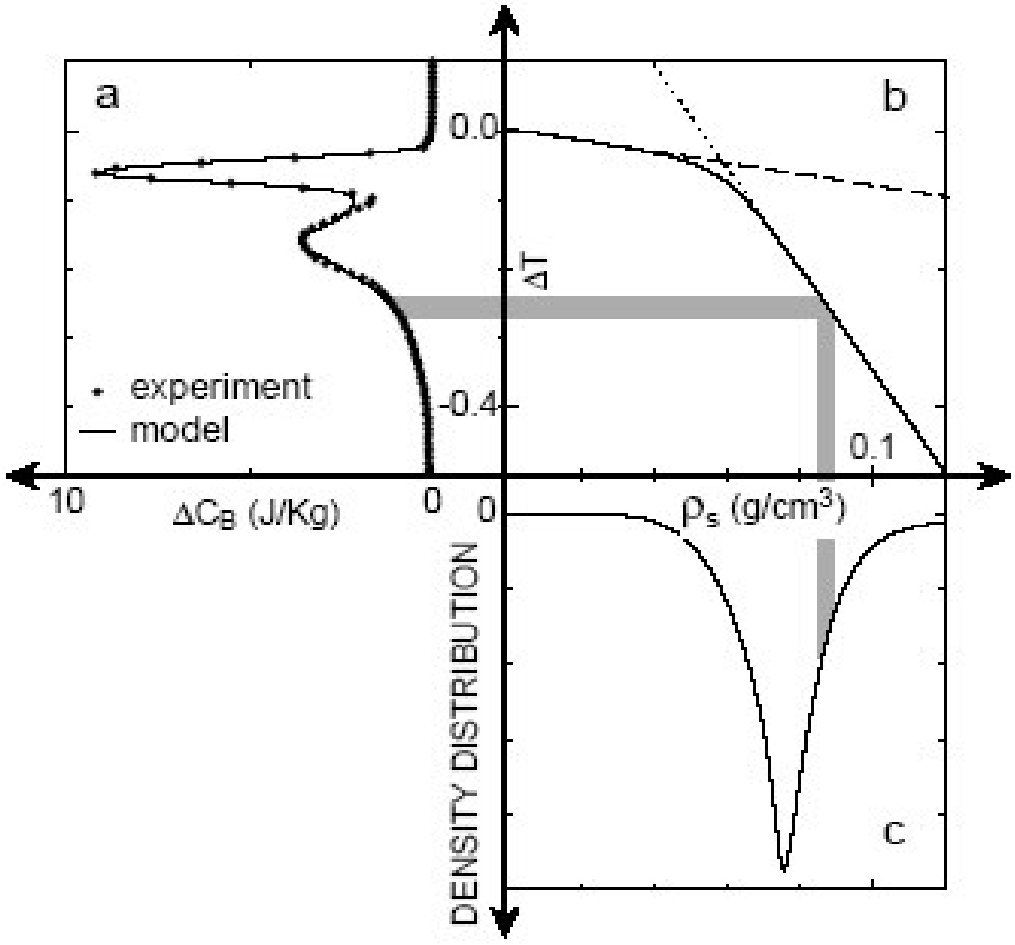


Figure 5.4: Proposed interpretation. Panel (a): Model (continuous line) and ΔC_B (full dots) versus ΔT for the ρ_S = 0.075 sample. Panel (b): distribution of transition temperatures ΔT within the two-phase coexistence region (solid line) versus disorder density (ρ_S) crossing over from random dilution effects (dashed line) to random field effects (dotted line). Panel (c): Local density distribution versus disorder density which successfully reconstructs ΔC_B .

However, as the data indicates, the disorder distribution is not bimodal and a stronger coupling appears at lower T, resulting in stronger distortions and the appearance of topological defects, a situation of random orientational pinning not accounted for by random dilution. We suggest that this cross-over is the result of the progressive thinning of a sheath of disordered LC, which screens - for a penetration length - the silica surface. As the temperature is lowered further, the disordered LC surface layer converts to a defected nematic and the random dilution turns into a random field coupling, described by some other, certainly steeper, distribution of $\Delta T_{IN}(\rho_S)$ behavior (dotted line in Figure 5.4b). This cross – over is proposed here to be the origin of the observed double peak. A simple disorder density distribution (Figure 5.4c) transforms, by virtue of the transfer function in Figure 5.4b, into a double peaked $\frac{d[\phi_N(T)]}{dT}$ (line in Figure 5.4a).

In this approach, a rather featureless disorder distribution and a simple transfer function were chosen so as to yield an overlap with the ΔC_B (0.075) data, demonstrating that these few ingredients can reproduce the observed doubled transition. This model does not address the observed [9] complex $T_{IN}(\rho_S)$ shifts, which require a more detailed study. However, no liquid-crystal specific properties were invoked in this interpretation and, so, we speculate that this effect may be generic for first-order phase transitions with weak quenched random disorder.

References:

[1] M. Chan, N. Mulders, and J. Reppy, Physics Today 49, 30 (1996), and references therein.

[2] T. Bellini, L. Radzihovsky, J. Toner, and N. A. Clark, Science 294, 1074 (2001), and references therein.

[3] T. Bellini et al., Phys. Rev. Lett. 85, 1008 (2000); ibid 88, 245506 (2002)

[4] R. L. Leheny et al., Phys. Rev. E 67, 011708 (2003).

[5] G. S. Iannacchione et al., Phys. Rev. E 67, 011709 (2003).

[6] P. S. Clegg et al., Phys. Rev. E 67, 021703 (2003).

[7] Y. Imry and M. Wortis, Phys. Rev. B 19, 3580 (1979).

[8] A. Roshi et al., Phys. Rev. E 69, 031703 (2004).

[9] G. S. Iannacchione et al., Phys. Rev. E 58, 5966 (1998).

[10] P. Jamee, G. Pitsi, and J. Thoen, Phys. Rev. E 66, 021707 (2002).

[11] P. G. de Gennes and J. Prost, The Physics of Liquid Crystals (Clarendon Press, Oxford, UK, 1993), 2nd ed.

[12] A. Zywocinski, J. Phys. Chem. B 107, 9491 (2003).

[13] T. Bellini et al., Phys. Rev. E 57, 2996 (1998).

[14] Z. Li and O. D. Lavrentovich, Phys. Rev. Lett. 73, 280 (1994).

[15] I. Chirtoc, M. Chirtoc, C. Glorieux, and J. Thoen, Liq. Cryst. 31, 229 (2004).

[16] H. Yao and C. W. Garland, Rev. Sci. Instrum. 69, 172 (1998).

[17] M. Caggioni, T. Bellini, S. Barjami, F. Mantegazza, A. Roshi, and G. Iannacchione, in progress.

[18] T. Jin and D. Finotello, Phys. Rev. Lett. 86, 818 (2001).

[19] T. Bellini et al., Phys. Rev. Lett. 91, 085704 (2003).

[20] T. Bellini, N. A. Clark, and D. R. Link, J. of Phys.: Cond. Matt. 15, S175 (2003).

CHAPTER 6

CONCLUDING REMARKS

We have described the development of the new modulation calorimetry technique using RF-Field heating as the only source for heating the sample. Advantages of the technique include elimination of the temperature gradients across the sample leading to a higher precision in evaluating the heat capacity of the sample compared to the previous ac techniques. A frequency scan to determine the operating frequency region for our calorimeter has been carried out on a octylcyanobiphenyl (8CB) + 70-$\overset{0}{A}$-diameter silica spheres (aerosil) sample with a density of ρ_S = 0.1 g/cm^3, showing a wide plateau, which indicates the region where the heat capacity will be frequency independent. A temperature scan was then performed on this sample where first order nematic to isotropic and second order smectic-A to nematic transitions were shown to be consistent with the previous work. A signature in the phase shift for these transitions was also observed. A shift in the baseline of C^*/P_0 was reported due to the strong temperature dependence of the power applied to the sample using RF-field heating. The amplitude of the power P_0 applied to the sample, as shown from 2.28 in the theory chapter of this work, depends on the permittivity $|\varepsilon|$ and the loss factor (which is the ratio of the

imaginary and real part of the permittivity) of the sample. The permittivity of a dielectric material has a strong temperature dependence, and this shows clearly in our Figure 4.9 data, thus giving us the opportunity to investigate both the heat capacity and permittivity dependence from the temperature of the sample. This is a new and exciting feature of our new AC calorimeter by radio frequency field heating.

The heat capacity measurements with a relative resolution of better than 0.06%, and the phase shift changes with a resolution of 0.03%, were shown to be one of the advantages of our new modulation calorimetry technique using RF-field heating.

We then presented Radio Frequency calorimetry results on 8CB bulk and 8CB + aerosil dispersions. For the bulk 8CB, the step-like character of smectic-A to nematic transition, and a suppressed first order nematic to isotropic transitions (different from the traditional AC calorimetry data) indicated the strong dominance of the permittivity $|\varepsilon|$ and the loss factor $\tan\delta = \frac{\varepsilon_2}{\varepsilon_1}$ of the material. For the 8CB + aerosil samples at different silica density, our data were consistent with the previous work done using traditional AC calorimetry. The coupling between the smectic A and nematic phases decreases with increasing silica density. The evolution of the smectic A to nematic phase transition also indicates that the nematic susceptibility is being suppressed with increasing silica density.

In this work we presented an experimental study of LC+A samples made with a low birefringence material, 4' – transbutyl – 4 – cyano – 4 – heptyl - bicyclohexane (CCN47), in which the biphenyls group is replaced by saturated hydrocarbon analogs, yielding a birefringence (Δn) about 1/10 that of cyanobiphenyls. The use of such a material enables a better optical characterization both through the study of I(q) in the

single scattering regime - where the nematic (director) correlation length ξ_N can be extracted - and through the measurement of $\tau(T)$ in the Rayleigh-Gans regime, in which the turbidity is a simple function of the relevant properties of the local nematic, namely $\tau(T) \propto \phi_N(T)\Delta n^2(T)\xi_N(T)$ [13] where ϕ_N is the nematic volume fraction. We have thus undertaken a combined T-dependent optical and calorimetric investigation of CCN47 LC+A samples through the I-N transition over a range of silica densities displaying the double I-N transition peak. This work offers compelling evidence that the I-N transition with weak quenched random disorder proceeds via a two-step process in which random-dilution is followed by random-field interactions on cooling from the isotropic phase, a previously unrecognized phenomena.

APPENDIX

CALORRF CONTROL PROGRAM

```
/************************  RF_MAIN.cpp ********************************
This is the Main Thread of the RF_ALL.exe.
RF.EXE controls the RF Calorimetry experiment on calorrf.
It Measures the Heat Capacity of the sample for different temperature scans
Ramping, Stepping, or Time Scans (sit at the same temperature).
It comunicates through a GPIB interface with :
Temperature Controller Model:     Lakeshore 340
DMM  Model:          Keithley, 2002/Mem 2 option, with 2001-SCAN Scanner Card
Arbitrary Waveform Generator Model:    DS345 from Stanford Research Systems
Arbitrary Waveform Generator Model:    DS335 from Stanford Research Systems
To understand this program some knowledge of the AC Calorimetry is needed.
This program is multithreaded.
The RF_main Thread does the Cp measurement, Data saving in files and printing
on screen.
The Temperature control Thread  controls the temperature.
Both communicate through signals in order to synchronize the data taking thread
with the temperature control thread.
ACP_main calls the functions that do the following:
1-      Read the initialization variables from ini file                Read_ini(..)
2-      Check the input for errors                                     Check_error_input(..)
3-      Initialize the thread variables for Multithreading       Ini_Thread_Var(..)
4-      Check for accidental erasure of existing files           Check_file_exist(..)
5-      Calculate the approximate time for the experiment to finish    Exp_time(..)
6-      Set up the voltage output from DS345 and DS335                 Setds345()
7-      Write headers to files .dat and .par                           Write_Header(..)
8-      Calculate the Npts for full wave digitizing                    NPTS(..)
After, the Temperature Controling thread is created.
This function continues with data taking and logging.(inner do while loop)
The data taking involves:
0-      Reset or initialize at the begining of the loops
1-      Read the prt before digitizing                                 ReadPRT(..)
2-      Digitize                                                       Digitize(..)
```

3- Read the prt after digitizing ReadPRT(..)
4- Calculate Cp and other data Calc_Cp (..)
5- Save data to file Save_Data(..)
6- Print data on the screen
7- Save the raw data if Chisq is 'big' Save_Raw_Data (..)
8- check the status of the signals and decide wheather to continue digitizing or not.

Important part of this program is the acp_ramp.h file where most definitions and some varible declarations are located. This file needs to be included in the project. See Acp_Ramp.H for more info.

NOTE: ALL COMMENTS IN THE PROGRAM START WITH "//"

*** *Adopted for RF Calorimetry by SAIMIR BARJAMI on 1 August, 2004********

```
**********************************************************************
#include <math.h>
#include <time.h>
#include <dos.h>
#include <alloc.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include "ACP_Ramp.h"
 int                keith,
                    lake,
                    ds345;
 struct data        *rawdata;
 struct Equilb_     settle;
 struct Zone_       T_zone;
 DWORD              TempControl(LPDWORD Param);// Global definition for the ramping thread
 HANDLE             ZoneDone;                  // Signals when the n-th zone ramp is done
 HANDLE             RampDone;                  // Signals when the whole ramp is done
 HANDLE             StartDigitize;             // Signals the main thread when to start digitizing.
 HANDLE             T_Eqlbr_for_Step; HANDLE   Done_Digitizing;
// Signals the Stepping thread to move at the next temperature .
//**********************************************************************
DWORD main(void) adopted
{
 printf("\n\n\n\n\n\n");
 printf("\t\t*************************************************\n");
 printf("\t\t*                                               *\n");
 printf("\t\t*       Calor RF.  RF Calorimetry               *\n");
```

```
printf("\t\t*          RF Control Program                                    *\n");
printf("\t\t*                                                                *\n");
printf("\t\t******************************************************\n");
HANDLE              H_RampTemp;         // Handle to the RampTemp thread
DWORD               ThreadId;
 int     n=0,                    // Dummy indices
         flag,                   // flag==1 when some condition is satisfied
         flag1=0,                // flag==0 otherwise.
         ptno=0;                 // The point number.
 char    time[30];               // Hold the time from CPU.
 double    Cp[11],               // Array holding final Cp calculations.
         prt1,                   // First PRT reading before digitizing.
         prt2                    // Second PRT reading (after digitizing.
         t_prt1,                 // Temperature of PRT before digitizing.
         t_prt2,                 // Temperature of PRT after digitizing.
         tau,                    // Time for Keithley to read 1 point.
         par[NFUNC];            // Array to hold parameters from fit.
 //float          tot_time;
 struct FileName       name;               // Holds the names of the files
 struct Pow_           P;                  // Holds the power and phase at both
frequencies
 struct ACP_           acp;                // General struct to hold all setup
variables
//*********************************************************************
sleep(6);
system("CLS");
acp = Read_ini();                   // Read the initialization file
Check_error_input(acp);             // Checks all inputs from ini file for errors.
Ini_Thread_Var();                   // Initialize global thread handles.
acp.volt.freq = acp.volt.freq/1000;      // Change frequency from mHz to Hz
settle = acp.equlbr;                // Initialize settle needed for the ramp thread
T_zone = acp.zone;                  // Initialize T_zone needed for the ramp thread
stpcpy(name.parfile,acp.file.name);      // Copy the name of the file at name
stpcpy(name.datafile,acp.file.name);
strcat(name.datafile,".dat");                          //  Add the .DAT extension
to data file name
strcat(name.parfile,".par");                //  Add the .PAR extension to parameter
file name
Check_file_exist( name.datafile,acp.file.mode); // Check accidental erasure of file
system("CLS");
printf("\n\n\t\t         Experiment started\n");
acp.zone.tot_t = Exp_time( acp);            // Calculate run duration;write it at ini
file
setds345(acp.volt.base_freq, acp.volt.amp,acp.volt.offset,acp.volt.freq);          //
Setup DS345 Generator
Write_Header(name,acp);             // Write headers at the .dat and .par files
```

```
acp.digtz.npts = NPTS(acp, &tau);          // Find NPTS to digitize from 'n_heat_per' and tau
acp.digtz.tau =  tau;                      // Store tau at acp.digtz.tau .
//  Allocate memory for the struct rawdata in the farheap region.
rawdata   = (struct data *) farmalloc((acp.digtz.npts+acp.digtz.n_filt+1)*sizeof(struct data));
if (!rawdata)
{
printf("Memory allocation error for rawdata struct - aborting\n");
system("PAUSE");
exit(1);
}
P = Power(acp, name);                      // Measure the power applied to the cell.
H_RampTemp =  CreateThread (               // Start the child thread that handles the Equlibration and Ramping
NULL,                                      // Default security
0,                                         // Default stack size
(LPTHREAD_START_ROUTINE)TempControl,       // The name of the routine
NULL,
0,                                                 // Run thread immediately
&ThreadId );
if (H_RampTemp == NULL)                    // Check that the handle is created.
{
printf("\nTemperature Control Thread cretion failed with error %x.Exiting Application\n",GetLastError() );
system("PAUSE");
return -1;
}
//**************  Start of the data taking loop through all zones***************
do
{  n=n+1;                                  // Increase the zone number.
Write_SubHeader(name, n);                  // Write subheader at .par file.
ResetEvent(Done_Digitizing);               // Reset the Done_Digitizing, signal.
WaitForSingleObject (StartDigitize,INFINITE);      // Wait INFINITE for the signal to start digitize.
do
{         flag = 0;
ptno = ptno+1;
ResetEvent(Done_Digitizing);
if                  (!                  strcmp(acp.zone.type[n],"STEP"))WaitForSingleObject (T_Eqlbr_for_Step,INFINITE);
prt1 = ReadPRT();                          // Read PRT before digitizing.
t_prt1= prtot(prt1);                       // Convert Resistance to Temperature .
Digitize(acp,rawdata,ch_Thrm);                     // DIGITIZE the thermistor resistance.
prt2=ReadPRT();                            // Read PRT after digitizing.
```

```
t_prt2= prtot(prt2);                    // Convert Resistance to Temperature.
Calc_Cp ( Cp,rawdata,P, par, acp);      // Calculate Cp, phase.... etc from rawdata and P .
Cp[10] = t_prt2-t_prt1;                 // Cp[10] is delta T = Tafter - Tbefore
Cp[0]= (t_prt1+t_prt2)/2.0;             // Cp[0] is average PRT Temperature.
_strtime(time);                         // Get time from CPU.
Save_Data( Cp,par,name,ptno);           // Save the data to files .dat and .par .
if ( (ptno-1)%10 ==0) printf("\n\n#  \tTprt(K)  dT(mK)  Tac(mk)  C*(mJ/K)  phi(%R)\tChisq  Time\n");
printf("\n%i\t%6.3f                    %.1f\t%.2f\t%.2lf\t%.2lf\t%.2f  %s",ptno,Cp[0],Cp[10]*1000,Cp[3]*1000,Cp[4]*1000,Cp[5]*100,Cp[9],time);
if (Cp[9]>acp.digtz.chisq) Save_Raw_Data (rawdata,par,ptno,Cp[9],acp);
 SetEvent(Done_Digitizing);             // Signal the Temperature Contro thread that Digitizing has finnished
//  Check the status of signal ZoneDone. If this zone temperature has reached the
//  end Temperature i.e. signal ZoneDone is set, then get out of this zone (flag = 1 ).
if (WaitForSingleObject (ZoneDone,1000) == WAIT_OBJECT_0)       flag=1;
}while (flag == 0);
// End of inside one ramp zone loop

// Check if the whole Temperature Control (over all zones) is done, i.e the RampDone signal is set
if (WaitForSingleObject (RampDone,500) == WAIT_OBJECT_0)        flag1=1;
 }while  (flag1 == 0);
// End of all ramping zones loop
  // Close Handles for the RampTemp and the RampDone signaling.
  CloseHandle (H_RampTemp);
  CloseHandle (RampDone);
  CloseHandle (ZoneDone);
  CloseHandle (StartDigitize);
  // Free up the memory for rawdata
  if (rawdata)  farfree(rawdata);
  // Take devices off-line, DS345 and DS335 are already taken off-line in Setds345 function
  ibonl(keith,0);
  ibonl(BOARD,0);

  return 0;
}
```

/**********setds345.cpp **********

* This function sets the DS345 and DS335 to generate a high frequency sine wave modulated in amplitude, with specified frequency, Peak To Peak amplitude, and offset from zero.

```
*  The frequency , amplitude and offset are passed from the calling
*  routine. Before using them it checks the variables for the proper
*  limits.
*****************************************************************************/
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include "acp_ramp.h"
//***************************************************************************
void setds345(double v_freq,double v_amp,double v_offset,double mod_freq)
{

int  ds345, ds335;
v_freq,                          //Voltage frequency is returned in Hz//
v_amp;                           // Voltage amplitude
char       sinewave[50];         // String command to be sent to HP
char     cmd[512];
*****************************************************************************
// FIND GPIB DEVICE ADDRESS of DS345
ds345 = ibdev(BOARD, PAD_OF_DS345, NO_SAD, TIMEOUT, EOTMODE,
EOSMODE);
if (ibsta & ERR) gpiberr("Can't find DS345 online",ds345);
 // FIND GPIB DEVICE ADDRESS of DS345
ds335 = ibdev(BOARD, PAD_OF_DS335, NO_SAD,
TIMEOUT, EOTMODE, EOSMODE);
if (ibsta & ERR) gpiberr("Can't find DS335 online",ds335);
gpibwrt(ds335,"*rst\r\n");                    // Clear DS335
gpibwrt(ds335,"*cls\r\n");                    // Clear DS335
gpibwrt(ds345,"*cls\r\n");                    // Clear DS345
gpibwrt(ds345,"*rst\r\n");                    // Clear DS345
printf("\nSetting up  Stanford Research Function Generator....");
sprintf(cmd, "MENA0;FUNC0;FREQ%f\n", v_freq);
gpibwrt(ds345, cmd);
// Amplitude and offset of the wave
sprintf(cmd, "AMPL %fVP", v_amp);
gpibwrt(ds345, cmd);
sprintf(cmd, "OFFS %f", v_offset);
gpibwrt(ds345, cmd);
sleep(1);
printf("\nModulation ON....");
// Make sure modulation is off untill after loading
// Set the modulation type to AM (internal)
// Set the modulation waveform to SINE wave
```

```
sprintf(cmd, "MENA0;MTYP2;MDWF3\n");
gpibwrt(ds345, cmd);
// Set the AM modulation depth
sprintf(cmd, "DPTH 100\n");
gpibwrt(ds345, cmd);
// Set the modulation rate
sprintf(cmd, "RATE %f\n", mod_freq);
gpibwrt(ds345, cmd);
// Turn the modulation on
sprintf(cmd, "MENA1\n");
gpibwrt(ds345, cmd);

// Setting up DS335 Function Generator
sprintf(cmd, "FUNC0;FREQ%f;AMPL4VP;OFFS%f;KEYS1\n", mod_freq, v_offset);
gpibwrt(ds335, cmd);
//sprintf(cmd, "FUNC0;FREQ%f;AMPL2.8VP;OFFS1.2;KEYS1\n", mod_freq);
//gpibwrt(ds335, cmd);
// Display the message that the setup of the function generator is done
printf(" .. DONE.");
ibonl(ds345,0);                                    // Take DS345 offline.
ibonl(ds335,0);                                    // Take DS345 offline.
return;
}
//CALC_CP.cpp
// This function calculates the values for C*, phi, Tac at heat and voltage          *
// frequencies. It calls sinfit.cpp to fit the raw data and get from it the          *
// amplitudes of resistance oscillations from which calculates Delta R.
// Then changes it to to Tac and calculates Cp = P_w / (w*Tac) at w and 2w
// It gets passed the structure data, Pow_, and ACP_
// Returns data and par variables in parameter list that hold C* data and
// fitting parameters respectively.
//Adopted for calorrf by Saimir Barjami   ON    1 August, 2004
#include <math.h>
#include "ACP_Ramp.h"
// input parameters and returns in the parameter list the array of data and fitting
// paramaters par
void Calc_Cp (double data[10],struct data *rawdata,struct Pow_ P, double
par[NFUNC],struct ACP_ acp)
{
 double         omega1,             // Voltage Frequency in radians
                omega2,             // Power Frequency in radians
                DeltaR_2w,
                DeltaR_w,
                dTdR,              // dT/dR
                a[3];              // Coeficients of temperature convertion for thermistor
```

```
// Assign the thermistor coefficients to the array a[3].
 a[0] = acp.th.a0;
a[1] = acp.th.a1;
a[2] = acp.th.a2;
// Change the frequency from Hz to radians.This is passed to fit the voltage for power measurement
omega1=2*pi*acp.volt.freq;
omega2=2*omega1;
//  Fit the digitized thermistor resistance
data[9] = sinfit(acp.digtz.npts,rawdata,par,omega1);            // data[9] = chisq
data[2]
data[1] = trtot(data[2],a);                                                  // Convert R_avg to Temperature Tth_avg
// Find the dT/dR (derivative of T wrt R)
dTdR= (1/data[2]) *(data[1]*data[1])*(a[1]+2.0*a[2]*log(data[2]/R0));
// Calculate Tac from Delta Rth then Cp and phi at the voltage frequency
DeltaR_w = sqrt(par[0]*par[0] + par[1]*par[1]);
data[3] = DeltaR_w * dTdR;                                  //          Tac_w = DeltaR_w * dTdR;
data[4] = P._w1/(omega1*data[3]);                    //    Cp_w = P._w1/(omega1*Tac_w);
data[5] = atan(par[1]/par[0])- P.phi_w1 + pi/2;         //          phi_w  =  atan(par[1]/par[0])- P.phi_w1 + pi/2;
// Calculate Tac from Delta Rth then Cp and phi at the heating frequency
DeltaR_2w = sqrt(par[2]*par[2] + par[3]*par[3]);
data[6] = DeltaR_2w * dTdR;                                 // Tac_2w = DeltaR_2w * dTdR;
data[7] = P._w2/(omega2*data[6]);                   // Cp_2w = P._w2/(omega2*Tac_2w);
data[8] = atan(par[3]/par[2]) - P.phi_w2;                 //  phi_2w  =  atan(par[3]/par[2])  - P.phi_w2;
 return;
}
//CHECKERR.CPP
//This function checks the input of the acp_ramp.exe (version 2) program
// The definnitions of the max and min  values are at the ACP_RAMP.H file
//Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <math.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "ACP_Ramp.h"
void Check_error_input (struct ACP_ acp)
{
int       flag =0,
          i;
// Check the file access setting
strupr(acp.file.mode);                                                                  //
change to upper case.
if (!(*acp.file.mode=='A'||*acp.file.mode=='W'))
```

```
{       printf("\nERROR IN THE FILE MODE. ALLOWED INPUT IS 'A' OR 'W' ");
flag=1;}
// Check the ramping parameters setting
for (i=1; i<=acp.zone.number; i++)
{       if(!(
acp.zone.start[i]>=T_min&&acp.zone.start[i]<=T_max&&acp.zone.end[i]>T_min&&ac
p.zone.end[i]< T_max||acp.zone.end[i]==-1))
{       printf("\n\nERROR IN THE TEMPERATURE INPUTS. ALLOWED INPUT IS
BETWEEN (%i, %i)K",T_min, T_max);
printf("\nError was found at line %i of temperature ranges",i);
flag =1; }
if(!( fabs(acp.zone.rate[i])>rate_min&&fabs(acp.zone.rate[i])<=rate_max ) )
{       printf("\n\nERROR IN THE RAMPING RATE INPUTS. ALLOWED INPUT IS
BETWEEN +/-(%i, %i]mK/h",rate_min, rate_max);
printf("\nTypical input is between 60 and 400 mK/h.");
flag =1; }
if ( strcmp(acp.zone.type[i],"TIME") && strcmp(acp.zone.type[i],"RAMP")&&
strcmp(acp.zone.type[i],"STEP") )
{       printf("\n\nERROR ZONETYPE IN ZONE %i. ALLOWED INPUT IS EITHER
'TIME' OR 'STEP' OR 'RAMP'",i);
flag=1;}
}
if (!(acp.digtz.n_heat_per >= Nper_min&&acp.digtz.n_heat_per <= Nper_max))
{       printf("\n\nERROR IN THE NUMBER OF HEATING PERIODS. ALLOWED
INPUT IS BETWEEN (%i , %i)  ",Nper_min, Nper_max);
printf("\nTypical input is 10.");
flag =1;}
// Check the digitizing parameters setting
if (!(acp.digtz.nplc >= nplc_min&&acp.digtz.nplc <= nplc_max))
{       printf("\n\nERROR IN THE NPLC NUMBER. ALLOWED INPUT IS
BETWEEN ( %.2f , %i )  ",nplc_min, nplc_max);
printf("\nTypical input is 10.");
flag =1;}
if (!(acp.digtz.n_filt >= nfilt_min&&acp.digtz.n_filt <= nfilt_max))
{       printf("\n\nERROR IN THE NUMBER OF POINTS TO FILTER. ALLOWED
INPUT IS BETWEEN (%i , %i)  ",nfilt_min, nfilt_max);
printf("\nTypical input is 12 with 'MOV' type filter.");
flag =1;}
strupr(acp.digtz.filt_type);                                                    //
change to upper case.
if ( strcmp(acp.digtz.filt_type,"REP") && strcmp(acp.digtz.filt_type,"MOV") )
{       printf("\n\nERROR IN THE FILTER TYPE SPECIFICATION. ALLOWED
INPUT IS EITHER 'REP' OR 'MOV' ");
printf("\nTypical input is 'MOV' with 12 points to filter.");
flag=1;}
// Check the voltage parameters setting
```

```
if(!(acp.volt.freq>=Vfreq_min&&acp.volt.freq<=Vfreq_max&&acp.volt.amp>=Vamp_
min&&acp.volt.amp<=Vamp_max ))
{       printf("\n\nERROR IN VOLTAGE RANGE INPUTS.");
printf("\nALLOWED RANGES ARE:\t VOLTAGE FREQUENCY (%imHz , %iHz)
",Vfreq_min,Vfreq_max/1000);
printf("\n\t\t\t VOLTAGE AMPLITUDE (%.2f , %i)V",Vamp_min,Vamp_max);
printf("\nTypical input is : Vfreq = 15.625mHz;   Vamp = 2.1V.");
flag=1;}
if                                                                                    (!(
fabs(acp.volt.offset)+acp.volt.amp/2.0<=V_MAX&&fabs(acp.volt.offset)<=2*acp.volt.a
mp ))
{       printf("\n\nERROR IN VOLTAGE OFFSET RANGE INPUT.");
printf("\nALLOWED RANGE IS SUCH THAT:\t |Voff| + Vamp/2 <= 10V");
printf("\n\t\t\t\t |Voff| <= 2* Vamp");
printf("\nTypical input is : Voff = 0.0V  i.e. no power at voltage frequency.");
printf("\nTO GET THE SAME Tac AT BOTH FREQUENCIES Voff ~ Vamp/8. ");
flag=1;}
// Check the temperature equilibration  parameters
if (!( acp.equlbr.time>=equlbr_min_time&&acp.equlbr.time<=equlbr_max_time ))
{       printf("\n\nERROR IN THE EQUILIBRATION TIME. ALLOWED INPUT IS
BETWEEN (%i , %i)  ",equlbr_min_time, equlbr_max_time);
flag =1;}
if                                                                                    (!(
acp.equlbr.deviation>=equlbr_min_deviation&&acp.equlbr.deviation<=equlbr_max_devi
ation ))
{       printf("\n\nERROR IN THE EQUILIBRATION BOUNDS. ALLOWED INPUT
IS BETWEEN (%i , %i) mK ",equlbr_min_deviation, equlbr_max_deviation);
flag =1;}
strupr(acp.equlbr.ctrloop);                                                           //
change to upper case.
if (!( *acp.equlbr.ctrloop=='A'||*acp.equlbr.ctrloop=='B' ))
{       printf("\n\nERROR IN THE SPECIFICATION OF TEMPERATURE
CONTROL LOOP. ALLOWED INPUT IS 'A' OR 'B' ");
flag=1;}
// Check the thermistor's (resistance to temperature) conversion coefficents
if(!(acp.th.a0>=a0_min&&acp.th.a0<=a0_max&&acp.th.a1>=a1_min&&acp.th.a1<=a1_
max&&acp.th.a2>=a2_min&&acp.th.a2<=a2_max))
{       printf("\n\nERROR IN THE THERMISTOR COEFFICIENTS INPUT.");
printf("\nALLOWED RANGES ARE BETWEEN:\t a(0) = (%.0f, %.0f)E-3
",a0_min*1e3,a0_max*1e3);
printf("\n\t\t\t\t a(1) = (%.0f, %.0f)E-3",a1_min*1e4,a1_max*1e4);
printf("\n\t\t\t\t a(2) = (%.0f, %.0f)E-3",a2_min*1e6,a2_max*1e6);
printf("\nTypical input is :  a(0)=2.36e-3,  a(1)=2.13e-4,  a(2)=2.5e-6 . ");
flag =1;}
if (flag==1)
```

```
{       printf("\n\n\nCHECK YOUR RF.INI FILE AND MAKE THE NECESSARY CHANGES.");
printf("\nAPPLICATION WILL TERMINATE .....\n");
printf("\n\n\n\nIf you are sure that your settings are not going to harm any equipment");
printf("\nyou may change the limits at the ACP_Ramp.h file and recompile the program.\n\n");
system("PAUSE");
exit(1);
}
return;
}
//*****************************CHECK.CPP*******************************
// Checks the existence of the file "filename" in the working directory;       *
// If this is true and the mode we are opening the file is 'w' i.e. everything *
// will be overwritten, it warns the user and exits the program if this was      *
// not intended . // Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <io.h>
#include <dos.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int Check_file_exist (char *filename,char *openmode)
{
char answer[10];
if ( (!access(filename,0)) && (*openmode=='w') )              // If filename exists... do the following
{  system("cls");
printf("\n\n\n\tWARNING !!!\n");
printf("\nThe file '%s' exists in this directory.\n\nAll it's contents will be erased!!",filename);
printf("\nYou might be loosing valuable data !! ");
printf("\n\nDO YOU INTEND TO DO THIS  (y/n)? ");
scanf("%s", answer);
strupr(answer);                                 // change answer to upper case
if ((*answer=='N'))
{       printf("\n The program will terminate.\n Change the name of the file or the mode and restart it (:|)");
sleep(8);
exit(0);
}
}
return 0;
/*********************cT_Main.CPP************************************
This is the Temperature Control Thread.                                 *
When it starts from the main thread it initializes LAKESHORE.           *
Then it starts the Ramp from the first zone to the last, by calling the     *
```

```
appropriate function that controls the temperature depending on the type    *
of the zone. At the and it signals the main thread that the whole ramp      *
is done and takes Lakeshore offline
If the zone is TIME  then  Time_scan(nzone) will control the temperature    *
If the zone is STEP  then  Step_T(nzone) will control the temperature       *
If the zone is RAMP  then  Ramp_T(nzone) will control the temperature               *
*****************************************************************************
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004.


*****************************************************************************
#include <time.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include "acp_ramp.h"
DWORD TempControl(LPDWORD Param)
{
void                                                    equilb(float,float,float);          //    Set
& equilibrate at the setpoint
extern int                                  lake;
extern HANDLE                                       RampDone;
extern struct Zone_     T_zone;
int         nzone;


// FIND GPIB DEVICE ADDRESS  of LAKESHORE
lake   =   ibdev(BOARD,   PAD_OF_LAKE,   NO_SAD,   TIMEOUT,   EOTMODE,
EOSMODE);
if (ibsta & ERR) gpiberr("Can't find LAKESHORE online",lake);
ibclr(lake);
gpibwrt(lake,"*cls\r\n");                                                   // Clear the Statusbyte
register
sleep(10);
//  Start the loop that traverses all zones
for  (nzone=1;nzone<=T_zone.number;nzone++)
{if (! strcmp(T_zone.type[nzone],"TIME"))Time_scan(nzone);
if (! strcmp(T_zone.type[nzone],"STEP"))Step_T(nzone);
if (! strcmp(T_zone.type[nzone],"RAMP"))Ramp_T(nzone);
}                               // Here is the end of all zones (For loop)
SetEvent(RampDone);                 // Signal the main thread that the whole ramp is done

ibclr(lake);
```

```
ibonl(lake,0);
return 0;

/**********************CTRL_T.CPP************************************
This file contains the individual Temperature control functions:          *
For TIME type zone  : void  Time_scan(int no)                                    For
STEP type zone  :       void  Step_T(int no)
For Ramp type zone  : void  Ramp_t(int no)
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004*
*************************************************************************
#include <time.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include "acp_ramp.h"
//*************************************************************************
//This Function handles the temperature control for a TIME type temperature zone;
// It sets the temperature at the initial temperature for the zone and than
//Sleeps for (3600 * T_zone.rate[no])  seconds. The input T_zone.rate[no] is in
//hours.
void  Time_scan(int no)
{
void                                                    equilb(float,float,float);
                        // Set & equilibrate at the setpoint
extern int                              lake;
extern HANDLE                                   ZoneDone;
extern HANDLE                                   StartDigitize;
extern HANDLE                                   Done_Digitizing;
extern struct Equilb_   settle;
extern struct Zone_     T_zone;


char                            cset[50];
                                                // Command to set the control loop
float                           setpoint;
printf("\nTime scan at %.3f for %.2f hours",T_zone.start[no],T_zone.rate[no]);
setpoint=T_zone.start[no];                              // Set temperature at the starting T
sprintf(cset,"CSET 1,%s,1,1,1\r\n",settle.ctrloop);
gpibwrt(lake,cset);
                                        // Set the control loop 1 for T control
// EQUILIBRATE at the starting temperature within +/- settle.deviation for settle.time
sec .
equilb(setpoint,settle.deviation/1000.0,settle.time);
```

```
SetEvent(StartDigitize);
// Signal the main thread to start digitizing.
// Do nothing but Sleep for T_zone.rate[no] hours.The temperature is controlled from the
Lakeshore
sleep(3600 * T_zone.rate[no]);                          // Just sleep for 'T_zone.rate[no]'
hours
printf("\nWaiting for the last digitizing to end");
WaitForSingleObject (Done_Digitizing,INFINITE);                  // Wait for the
last ongoing digitizing to end.
printf("\nTime Zone # %i is DONE", no);                // This zone is done .
SetEvent(ZoneDone);
}
//*****************************************************************
//This Function Handles the temperature control for a STEP type temperature zone;
// This function steos the temperature of the bath from the starting temperature
// of the zone no 'T_zone.start[no]' to the ending temperature of the zone no
// 'T_zone.end[no]' with steps of 'T_zone.rate[no] mK'
void  Step_T(int no)
{
void                                        equilb(float,float,float);      //    Set
& equilibrate at the setpoint
extern int                          lake;           // Lakeshore ID .
extern HANDLE                       ZoneDone;
extern HANDLE                       StartDigitize;
extern HANDLE                       T_Eqlbr_for_Step;
extern HANDLE                       Done_Digitizing;
extern struct Equilb_   settle;
extern struct Zone_     T_zone;
int                         flag=0,                 // Flag to check that ramp is
set OK
i=0;
char       newsetp[40],          // Command to set the new setpoint
cset[50];
// Command to set the control loop
float                       setpoint;
printf("\nStep      Zone      from      %.3f      to      %.3f      with      %.0f      mK
step",T_zone.start[no],T_zone.end[no],T_zone.rate[no]);
sprintf(cset,"CSET 1,%s,1,1,1\r\n",settle.ctrloop);
gpibwrt(lake,cset);                                     // Set the control loop 1
gpibwrt(lake,"PID 1, 800, 99, 20\r\n");
// Set PID .
SetEvent(StartDigitize);                        // Signal the main thread to start
Digitizing
do{      setpoint = T_zone.start[no] + i*T_zone.rate[no]/1000;                  //
Calculate what the setpoint should be .
sprintf(newsetp,"SETP 1, %.4f\r\n", setpoint);
```

```
gpibwrt(lake,newsetp);
// Set lakeshore at that setpoint
// EQUILIBRATE at the starting temperature within +/- settle.deviation for settle.time sec .
equilb(setpoint,settle.deviation/1000.0,settle.time);
//        The main thread will wait until the equilibrium is set to begin digitizing.
SetEvent(T_Eqlbr_for_Step);
// Signal the main thread to digitize.
WaitForSingleObject (Done_Digitizing,INFINITE);            // Wait here until digitizing is done
// then it can go to next Temperature.
i=i+1;
// Check if the ramp is done by checking that the setpoint  is < endtemp
// Calculate the next setpoint and check for flags....
setpoint = T_zone.start[no] + i*T_zone.rate[no]/1000;
if (((T_zone.rate[no]> 0) && (setpoint > T_zone.end[no])) ||  ((T_zone.rate[no]< 0) && (setpoint < T_zone.end[no])))
{                  flag = 1; }                                        // Get out of the do_while
}while (flag == 0);
printf("\nStepping Zone # %i is DONE", no);
SetEvent(ZoneDone);                                                  // Signal the main thread that the n-th ramp is done.
return;
}


//*********************************************************************************************************
//        This Function Handles the temperature control for a RAMP type temperature zone;
// It equilibrates firstly at the initial temperature of the zone and than
// starts the temperature ramp.(Gets the initial time, Calcualates how much
// time has lapsed from the beggining of ramp, calculates what the temperature
// should be, and sends this temperature for the lakeshore to equilibrate.
// Because the setpoint is changed every ~ 1sec, the effective behavior is to
// ramp the bath temperature, since the bath is not given time to equilibrate
// at the setpoint and the latter is changed "very fast". The choice of PID has
// been proven to work good for smoother Temperature Ramp for the current setup
// of CalorA and CalorB and for different ramp rates.
void  Ramp_T(int no)
{
void                                                  equilb(float,float,float);        // Set & equilibrate at the setpoint
extern int                                lake;
extern HANDLE                                   ZoneDone;
extern HANDLE                                   StartDigitize;
```

```
extern HANDLE                   Done_Digitizing;
extern struct Equilb_  settle;
extern struct Zone_    T_zone;
int        flag,                    // Flag to check that ramp is set OK
           eqlbr_interval = 20;          // Just 20s equilibration interval before
ramping
char       newsetp[40],          // Command to set the new setpoint
cset[50];                        // Command to set the control loop
clock_t                  start_time,
time_now;
float      t,
           setpoint,
           eqlbr_bound = 5.0;          // Just 5mK equilibration bounds before ramping
           printf("\nRamp Zone from %.3f to %.3f at %.0f mK/h ramp
rate",T_zone.start[no],T_zone.end[no],T_zone.rate[no]);
setpoint=T_zone.start[no];
sprintf(cset,"CSET 1,%s,1,1,1\r\n",settle.ctrloop);
gpibwrt(lake,cset);                                    // Set the control loop 1
// EQUILIBRATE at the starting temperature within +/- settle.deviation for settle.time
sec .
equilb(setpoint,eqlbr_bound/1000.0,eqlbr_interval);
SetEvent(StartDigitize);                                                    //
Signal the main thread to start digitizing.
flag=0;
// Reset the value of the flag=0 .
gpibwrt(lake,"PID 1, 30, 99, 20\r\n");
// Set PID to 30,99,20 .
start_time = clock();
// Get the start time (read it from the CPU) .
do{      time_now = clock();
// Get the now time .
t = (time_now - start_time)/CLK_TCK;                                        // Find
the lapsed time.
setpoint = T_zone.start[no] + (T_zone.rate[no])/(3.6e6)*t;   // Calculate what the setpoint
should be .
sprintf(newsetp,"SETP 1, %.4f\r\n", setpoint);
gpibwrt(lake,newsetp);
// Set lakeshore at that setpoint
sleep(1);
// Check if the ramp is done by checking that the setpoint  is < endtemp
if (((T_zone.rate[no]> 0) && (setpoint >= T_zone.end[no])) ||  ((T_zone.rate[no]< 0) &&
(setpoint <= T_zone.end[no])))
{                flag = 1;}                          // Get out of the do_while
}while (flag == 0);
printf("\nWaiting for the last digitizing to end");
```

```
WaitForSingleObject (Done_Digitizing,INFINITE);                    // Wait
for the last ongoing digitizing to end.
printf("\nRamping Zone # %i is DONE", no);
SetEvent(ZoneDone);
// Signal the main thread that the n-th ramp is done.
return;
}
//DIGITIZE.CPP
//THIS FUNCTION DIGITIZES THE VOLTAGE OF CHANNEL 3,4 OR 4-W
RESISTANCE                *
//ON    CHANNEL 2 WHERE THE THERMISTOR IS CONNECTED.
// It gets the ACP_ structure and the channel # to digitize .
// It returns the structure data with raw data.
// Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <dos.h>
#include <alloc.h>
#include <stdio.h>
#include <windows.h>
#include "decl-32.h"
#include "ACP_Ramp.h"
void Digitize(struct ACP_ acp,struct data *p_raw, int chann)
{
 FILE                                    *fptemp;
  extern  int                          keith;
// GP-IB address of Keithley 2002
int                                    i,n,
// Dummy indices
stb_value,                                                          // Status Byte
Value
n_filt = acp.digtz.n_filt,
ch_range = 2000000,
npts = acp.digtz.npts + n_filt,
period = int(1.0/acp.volt.freq),
other_t = 32 + int(npts*0.008),              // 32s == time to measure twice the prt +
time to setup;

// + time to Download npts points from Keithley .
sleep_t = int(npts* acp.digtz.tau) + period - other_t%period;  // Time to sleep while
Keithley is digitizing.
char    buffstore[100],
        trig[100],
        range[100],
        filter[100],
        func[100],
        scan[100],
        SP_r[10],
```

```
        *p,                                     // Dummy Variable pointer
        *keidat;                                // String where Data are buffered from
KEITH.
double          res,                    // Variables where the stripped data from the string
tim,                                    // keidat will be stored temporarily before printing
// them to the file temp.dat
R[6100],
t[6100];
ibclr(keith);
static char
resetctrl[]="*rst;\r\n",                                        // Reset
controls in INIT, ARM,
resetst[]=":stat:pres;*cls;\r\n",                               // Reset status byte
statenb[]=":stat:meas:enab 512;\r\n",                           // Enable BFL(buffer full bit
msbenb[]="*sre 1;\r\n",                                         // Enable MSB
( Master Status Byte)
buffcl[]=":trac:cle;\r\n",                                      // Clear
buffer
arm[]=":arm:lay2:sour ext;\r\n",                                // Set trigger
layer 2 to external
dataformat[]=":form:elem read,time;\r\n",                       // Format for downloading
over GPIB
dumpdata[]=":trac:data?;\r\n",                                  // Prepare
keithley to send data
init[]=":init\r\n",
// Initiate everything
azeroff[]=":syst:azer:stat 0\r\n";                              // Turn Autozero OFF

sprintf(func,"fres");
 if (chann > 2)
 {      ch_range = 2;
n_filt= 1;
npts = acp.digtz.npts;
sprintf(func,"volt:dc"); }
// Set channel 'chann' to measure 'func'
sprintf(scan,"ROUTE:CLOSE (@ %i);:func '%s';\r\n",chann, func);
// set range nplc and digits of resolution
sprintf(range,":%s:range %i;nplc %.2f;digits 9\r\n",func,ch_range,acp.digtz.nplc);
// set storage format as full and npts
sprintf(buffstore,"trac:poin %4d;egr full;feed sens1;feed:cont next;\r\n",npts);
//set trigger for layer 1
sprintf(trig,"trig:COUNT %4d;\r\n",npts);
  // set number of points to filter,and  filter type. If filter <=1 do not filter
  if (n_filt <=1)
  {     sprintf(filter,":fres:aver OFF;\r\n");
                n_filt = 1;
```

```
  }
  else    sprintf(filter,":%s:aver                    ON;aver:tcon                    %s;:%s:aver:count
%i;\r\n",func,acp.digtz.filt_type,func,n_filt);
  //  Allocate memory for the variable keidat in the farheap region.
  keidat = (char *) farmalloc(4000000);
  if (!keidat)
{
 printf("Memory allocation error for keidat - aborting\n");
exit(1);
}
// WRITE ALL THE COMMANDS TO KEITHLEY
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  gpibwrt(keith,statenb);
  gpibwrt(keith,msbenb);
  gpibwrt(keith,buffcl);
  gpibwrt(keith,range);
  gpibwrt(keith,scan);
  gpibwrt(keith,filter);
  gpibwrt(keith,arm);
  gpibwrt(keith,trig);
  gpibwrt(keith,buffstore);
  gpibwrt(keith,azeroff);
  gpibwrt(keith,init);                                          /* Start everything  */
// The next line waits Keithley to assert that the buffer is full
// i.e. the corresponding bit is set, and an SRQ is required
//  ibwait(keith,RQS);
  sleep(sleep_t);                                                       //  sleep  while
digitizing .
  do
  {
            sleep(1);
            gpibwrt(keith,"*STB?\r\n");
            sleep(1);
            ibrd(keith,keidat,9L);
            keidat[ibcnt]= '\0';
            stb_value = atoi(keidat);
           } while (stb_value != 65);

// Send the dataformat, and read from buffer into keidat.
  gpibwrt(keith,dataformat);
  gpibwrt(keith,dumpdata);
  ibrd(keith,keidat,4000000L);
// Null terminate keidat at the ibcnt character.
  keidat[ibcnt]= '\0';
//  Read from keidat the readings and write them at the file temp.dat
```

```
//  Time and resistance.
//  printf("\n Stripping keidat and writting at temp.dat.");
  p = keidat;
  i = 0;
  n = 0;
  fptemp=fopen("temp.dat","w");
  fprintf(fptemp,"Time\t\tR_Thermist\n");
// Read the first value from the String KEIDAT
  sscanf(p,"%lf",&res);
// Continue to read the rest of the data from KEIDAT.
  for ( ; *p != '\0'; ++p)
  {
          if (*p == ',')
          {       i++;
                                  if ( i%2!=0 )
                                  {           sscanf(p+1,"%lf", &tim);
                                              fprintf(fptemp,"%9.6f\t%f\n",tim,res);
                                              R[n] = res;
                                              t[n] = tim;
                                              n++;
                                  }
                                  else     sscanf(p+1,"%lf",&res);
          }
  }
  fclose(fptemp);
  ibrsp ( keith, SP_r );          // serial poll keithley to release SRQ
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
// Free up the memory location allocated before.
  if (keidat)  farfree(keidat);
// Discard the first n_filt points which are skewed from the filtering
// when giving values at p_raw structure.
  fptemp = fopen("temp1.dat", "w");
  for (n= n_filt -1; n<=npts-1; n++)
  {   fprintf(fptemp,"%9.6f\t%f\n",t[n],R[n]);
                  p_raw->value = R[n];
                  p_raw->time = t[n];
                  p_raw++;
  }
  fclose(fptemp);
return;
}
/* **********************  EQUILB.CPP  ********************************
*  This function equlibrates at the given setpoint within the
*  specified temperature bounds and interval.
*
```

```
*  It  gets from the calling function the setpoint,temperature bounds
*  and the interval for equilibrium via the variables setpnt, deviation,
*  interval. It clears LAKESHORE, sets the setpoint
*  and after making sure that the statusbyte is cleared sends the
*  Settle command to equilibrate at the setpoint for 'interval' seconds
*  within 'tempdev' degrees. The program checks for the equilibrium
*  by polling the settle bit (2) [weight==4].
// Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <dos.h>
#include <stdio.h>
#include <windows.h>
#include "decl-32.h"
void gpibwrt(int device, char command[512]);                    // Send command to
GPIB device
void gpiberr(char *msg, int);                                                    //
Display GPIB error
void equilb(float setpnt,float deviation,float interval)
{
// setpnt                                          Temperature setpoint for
equilibrum (K)
// deviation                              Allowable temperature deviation from
setpoint (mK)
// interval                                interval interval to test for stability
(Seconds)
 extern int       lake;
 int                          byteval;          // Status Byte Value
 char                         lakeout[50],      // Reading data buffer
                              newstp[40],              // String sent to LAKE to set the
NEWSETPOINT
equilb[40];         // String sent to LAKE to start the SETTLE routine
*************************************************************************
ibclr(lake);
 gpibwrt(lake,"*cls\r\n");                                      //       Clears       the
Statusbyte register
gpibwrt(lake,"PID 1, 800, 50, 20\r\n");    // Set the PID for equilibrium
sprintf(newstp,"SETP 1, %7.3f\r\n", setpnt);
 gpibwrt(lake,newstp);                                          // Set a new setpoint
//  Start the SETTLE routine with temperature deviation == deviation
//  and test equlibrium for== interval seconds.
            sprintf(equilb,"SETTLE %f,%.0f\r\n", deviation, interval);
            gpibwrt(lake,equilb);
            gpibwrt(lake,"settle?\r\n");
            sleep(1);
            ibrd(lake, lakeout,30L);
            sleep(1);
lakeout[ibcnt]= '\0';
```

```
// Read the values of interval & deviation from lakeout and print them on the screen
sscanf(lakeout,"%f,%f", &deviation, &interval);
printf("\nSETPOINT @ %7.3f K,(EQUILIB FOR %.0f sec, +/- %.1f mK ...",setpnt,interval,deviation*1000);
// The next loop makes sure that the status byte is cleared. Normally the *cls
// command would clear it but it seemed that this was not true always with the Lakeshore
// The loop takes care of this problem.
gpibwrt(lake, "*CLS\r\n");
do { //Sleep(100);
                        gpibwrt(lake,"*cls\r\n");
                        gpibwrt(lake,"*stb?\r\n");
                        //Sleep(100);
                        ibrd(lake, lakeout,6L);
                        lakeout[ibcnt]= '\0';
                        byteval = atoi(lakeout);
} while (byteval != 0);
sleep(1);
gpibwrt(lake, "*cls\r\n");
gpibwrt(lake, "*SRE 4\r\n");                                    // Enable settle bit [2] weight == 4.
// The next loop checks if the program is completed, i.e. the sample is at equilibrium T .
        do
{
//Sleep(5000);
//  Sleep for 5000 mSeconds
gpibwrt(lake,"*STB?\r\n");
//Sleep(100);
ibrd(lake,lakeout,4L);          //  Query status byte; if ==0 then settle
lakeout[ibcnt]= '\0';                                   //  program has not finished yet; if ==4 then
byteval = atoi(lakeout);                                //  program is done and sample has reached
equilibrium
} while (byteval != 4);

gpibwrt(lake, "*cls\r\n");
printf("\..DONE");
}
//Exp_Time.cpp
// Calculates the expected duration of experiment in hours.
// Adds at the ACP_ALL.INI file the date and time that experiment
// started and the calculated duration time of the experiment
// Adopted for calorrf by Saimir Barjami   on    1 August, 2004
//  Modiffied by          A. Roshi on 28 May 2003
//  Added the time scan calculation when acp.zone.end[i] == -1
//          Modified By A. Roshi on 5 June,2003.
//  Added the calculation for step type zone where equilibration is needed
//          before digitizing.
#include <time.h>
```

```
#include <dos.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "ACP_Ramp.h"
double Exp_time( struct ACP_ acp)
{
        double t_ = 1.0/6.0,            // Duration of power measurement (about 10min)
        time_per_step,                  // Time to equilbrate + digitize for step type zone
        data_time = (acp.digtz.n_heat_per + 2)/(2.0*acp.volt.freq), // Time to digitize in
sec
        eq_time = acp.equlbr.time,                      // Time to equlibrate T in sec
        n_steps;                                        // Number of steps for step
type zone or ,
// time to ramp within zone for ramp type zone
char time[30],
date[30];
FILE            *fpini;
int             i;
for (i=1; i<=acp.zone.number; i++)
{  // Time per 1 step in Hours == time of digitizing + time to equilibrate
// + extra time factor proportional to step size.
// From experiments the bigger the step change the bigger this extra time
// gets, thus to first approximation is linear with step size.
time_per_step = (data_time + eq_time + 1.7*acp.zone.rate[i])/3600;
n_steps = 1000.0*fabs((acp.zone.end[i]- acp.zone.start[i])/acp.zone.rate[i]);
if (! strcmp(acp.zone.type[i],"STEP"))
{       t_ = t_ + time_per_step * n_steps;
continue;}
if (! strcmp(acp.zone.type[i],"TIME")) t_ = t_ + acp.zone.rate[i];   // If time scan add
hours to wait
else    t_ = t_ + n_steps;
}
printf("\n Calculated experiment duration %.1f hours.",t_);
_strtime(time);
_strdate(date);
fpini=fopen("RF.ini","a");
fprintf(fpini,"\n***********************************************");
fprintf(fpini,"\nEXPERIMENT STARTED ON  %s at %s .",date,time);
fprintf(fpini,"\nCalculated experiment time ~ %.1f hours.",t_);
fclose(fpini);
return t_;
}
//**********************FIT.CPP*****************************************
//  This routine is taken from previous Acp Programms in GSI Lab.
```

```
// Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <io.h>
#include <math.h>
#defineX        (pd_local->time)
#defineY        (pd_local->value)
#include "ACP_Ramp.h"

extern PFD func_array[];
double matinv(int n_siz,double r[NFUNC][NFUNC],double rinv[NFUNC][NFUNC]);
//////////////////////////sinfit/////////////////////////////////////
double sinfit(int npts,struct data *pdata,double *ppar,double freq)
{int     i,j,k;                  /* i loops through the data points  */
/* j, k loop through the functions       */
static double                           /* Static becuase large arrays        */
siga[NFUNC],                /* sigma's for parameters.  */
/* Keep maybe use later*/
r[NFUNC][NFUNC], ry[NFUNC],
rinv[NFUNC][NFUNC],
s[NFUNC][NFUNC], sy[NFUNC];
double  chisq,
syy, ryy,
yavg, wavg,                     /* avg. values of y and weight=1/y         */
favg[NFUNC],                            /* average values of the functions  */
scr, scr1, scr2, scr3;
struct data
*pd_local;                      /* data pointer      */
#ifdef  DEBUG
FILE    *fd_tmp;
struct data *ddp;
int dint;
#endif
#ifdef  DEBUG
fd_tmp = fopen( "/tmp1/sf_tmp", "a");
fprintf( fd_tmp, "Sf_fit: npts = %d\n", npts);
fprintf( fd_tmp, "Sf_fit: freq = %g\n", freq);
fprintf( fd_tmp, "Sf_fit: pdata = %x\n", pdata);
fprintf( fd_tmp, "Sf_fit: value = %g, %lx\n", pdata->value,
 pdata->value);
printf( "Sf_fit: point = %ld %g\n", pdata->time,
pdata->value);
for( dint = 0, ddp = pdata; dint < 10; dint++, ddp++)
fprintf( fd_tmp, "\t%ld\t%lf\n", ddp->time, ddp->value);
fclose( fd_tmp);
#endif
//*****           Zero average values.  *****
for( j = 0; j < NFUNC; j++)
```

```
favg[j] = 0;yavg = 0;
//*****          Accumulate average values.  **********
#ifdef  DEBUG
printf( "\nSf_fit: Calculate avg. values.\n");
#endif
for( i = 0, pd_local = pdata; i < npts; i++, pd_local++){
yavg += Y;
for( j = 0; j < NFUNC; j++){
favg[j] += (*func_array[j])( (double)X, freq);
}
}
yavg /= npts;
for( j = 0; j < NFUNC; j++)
favg[j] /= npts;
//********                zero NFUNC stuff  ********
syy = ryy = 0;
for( j = 0; j < NFUNC; j++){
ppar[j] = siga[j] = ry[j] = sy[j] = 0;
for( k = 0; k < NFUNC; k++)
r[j][k] = rinv[j][k] = s[j][k] = 0;
}
//**********            calculate sigma's     ***********
#ifdef  DEBUG
printf( "\nSf_fit: Calculate sigma's\n");
#endif
for( i = 0, pd_local = pdata; i < npts; i++, pd_local++){
scr = Y - yavg;
syy += scr * scr;
for( j = 0; j < NFUNC; j++){
scr1 = (*func_array[j])( (double)X, freq) - favg[j];
sy[j] += scr * scr1;
s[j][j] += scr1 * scr1;
for( k = j+1; k < NFUNC; k++)
s[j][k] += scr1 *( (*func_array[k])( (double)X, freq) - favg[k]);
                }
        }
#ifdef  DEBUG
printf( "\nSf_fit: Done calculating sigma's\n");
printf( "\nSf_fit: Calculating r matrix\n");
#endif
//*******                calculate r matrix    ***********
for( j = 0; j < NFUNC; j++){
ry[j] = sy[j] / ( sqrt( s[j][j] * syy ));
for( k = j; k < NFUNC; k++){
        r[j][k] = r[k][j] = s[j][k] /
                                sqrt( s[j][j] * s[k][k] );
```

```
        }
}
//********                invert r matrix     ********************
#ifdef  DEBUG
        printf( "Sf_fit: Invert r matrix\n");
#endif
        if( matinv( NFUNC, r, rinv) == 0){
                static char a[] = "Matinv error\n";
write( 2, a, sizeof( a));
return( 0);
}


//************                calculate param's    ***************
for( j = 0; j < NFUNC; j++){
        ppar[j] = 0;
        for( k = 0; k < NFUNC; k++){
        ppar[j] += ry[k] * rinv[j][k];
        }
        ppar[j] *= sqrt( syy / s[j][j] );
        scr = npts - 1.;
        siga[j] = rinv[j][j] /(s[j][j] * scr);
        siga[j] = sqrt( siga[j] );
        }
ppar[NFUNC] = yavg;                     //  Calculate the constant term
// We want the average for thermistor temperature, not the constant part of
// the background.  K.S. 6-87
for( j = 0; j < NFUNC; j++)
ppar[NFUNC] -= ppar[j] * favg[j];
//*******                accumulate chi square     **************
#ifdef  DEBUG
printf("Sf_fit: param's = %lf, %lf, %lf\n", ppar[0], ppar[1], ppar[2]);
#endif
chisq = 0;
for( i = 0, pd_local = pdata; i < npts; i++, pd_local++){
scr = Y - ppar[NFUNC];
for( j = 0; j < NFUNC; j++){
scr -= ppar[j] * (*func_array[j])((double)X, freq);
}
chisq += scr * scr;
}
chisq /= npts - NFUNC - 1;
#ifdef  DEBUG
printf( "Sf_fit: Chisq = %g\n", chisq);
#endif
ppar[NFUNC]=yavg;
```

```
return( chisq);
}


//*                        matinv.cpp                                    *
#defineERR    0
#defineNOERR          1
#defineZETA  (1e-20)          //* Singularity criterion for inversion  *
#defineROW   0
#defineCOLUMN       1
#undef DEBUG
//**********            matrix inversion program
void swap(int n,double mat[][NFUNC],int flag,int first,int second);
double matinv(int n_siz,double r[NFUNC][NFUNC],double rinv[NFUNC][NFUNC])
{int
          i, j, k, l,                       /* loop indices        */
          ik[NFUNC], jk[NFUNC];             /* indices of max values  */
          double
          x[NFUNC],
          y[NFUNC],
          amax, save;
#ifdef  DEBUG
printf( "\nMatinv: n_siz = %d\n", n_siz);
#endif
if( n_siz == 2){
          double det;
          det = 1./(r[0][0] * r[1][1] -
          r[0][1] * r[1][0] );
          rinv[0][0] = r[1][1] * det;
          rinv[0][1] = -r[0][1] * det;
          rinv[1][0] = -r[1][0] * det;
          rinv[1][1] = r[0][0] * det;
          return( NOERR);
          }

//*********    copy r into rinv  ****
          for( i = 0; i < n_siz; i++)
          for( j = 0; j < n_siz; j++)
          rinv[i][j] = r[i][j];

//****  find the largest element in the submatrix  ***************************
for( k = 0; k < n_siz; k++){
          amax = 0.;
          for( i = k; i < n_siz; i++)
          for( j = k; j < n_siz; j++){
                    if( fabs( amax) < fabs( rinv[i][j]) ){
```

```
                                amax = rinv[i][j];
                                ik[k] = i;
                                jk[k] = j;
                                }
                        }
//*      interchange rows and columns to put amax in rinv[k][k]   ****************
                if( fabs( amax) < ZETA ){
#ifdef  DEBUG
printf( "Matinv: Singular matrix. i = %d, j = %d, pivot = %g\n",
ik[i], jk[j], amax);
#endif
return( ERR);
}
swap( n_siz, rinv, ROW, ik[k], k);
swap( n_siz, rinv, COLUMN, jk[k], k);
for( j = 0; j < n_siz; j++){
                        if( j != k){
                                double scr;
                                scr = -rinv[k][j];
                                x[j] = scr/amax;
                                y[j] = rinv[j][k];
                        } else {
                                x[j] = 1/ amax;
                                y[j] = 1;
                        }
                        rinv[k][j] = 0;
                        rinv[j][k] = 0;
                }
                for( i = 0; i < n_siz; i++){
                        for( j = 0; j < n_siz; j++){
                                double scr;
                                scr = y[i] * x[j];
                                rinv[i][j] += scr;
                        }
                }
        }
#ifdef  DEBUG
        printf( "Matinv: Before reordering");
        for( i = 0; i < NFUNC; i++){
                printf( "\n");
                for( j = 0; j < NFUNC; j++)
                        printf( "\t%g", rinv[i][j]);
        }
        printf( "\n");
#endif
//*********   restore ordering of the matrix    ********************************
```

```
        for( l = 0; l < n_siz; l++){
                k = n_siz - l - 1;
                swap( n_siz, rinv, COLUMN, ik[k], k);
                swap( n_siz, rinv, ROW, jk[k], k);
        }
#ifdef  DEBUG
printf( "\nMatinv: Done.\n");
#endif
return( NOERR);
}
/****SWAP FUNCTION*****/
void swap(int n,double mat[][NFUNC],int flag,int first,int second)
{
        int i;
        double scr;
        if( first == second)
                ;
        if( flag == ROW)
                for( i = 0; i < n; i++){
                        scr = mat[first][i];
                        mat[first][i] = mat[second][i];
                        mat[second][i] = scr;
                }
        else
                for( i = 0; i < n; i++){
                        scr = mat[i][first];
                        mat[i][first] = mat[i][second];
                        mat[i][second] = scr;
                }
}
//***************************   FIT_FUNC.CPP  ********  //
        Adopted for calorrf by Saimir Barjami   on    1 August, 2004
//  These are the FITTING FUNCTIONS
// Change these and the appropriate section at ACP_RAMP.H if you want to change
// the fit.
#include <math.h>
#include "acp_ramp.h"
        double sin1( double arg, double fpar){
                return( sin( fpar * arg));
        }
        double cos1( double arg,double fpar){
                return( cos( fpar * arg));
        }
        double sin2(double arg,double fpar){
                return( sin( 2.0 * fpar * arg));
        }
```

```
        double cos2(double arg,double fpar){
                return( cos( 2.0 * fpar * arg));
        }
        double x1(double arg,double){
                return( arg);
        }
        double x2(double arg,double){
                return( arg *arg);
        }
        double x3(double arg,double){
                return( arg * arg * arg);
        }
// Decclare the  Array of Pointers to Functions Returning Doubles
PFD func_array[] =
{               sin1,
                cos1,
                sin2,
                cos2,
                x1,
                x2,
                x3,
};
//  ************************   GPIBWRT.CPP   **********
//   This function sends to specified device the specified command
//   If for some reason it's impossible to send the command it calls the
//   gpiberr function which will find the reason  print it on the screen,
//   take the device off_line and terminate the program.
//  Adopted for calorrf by Saimir Barjami   on    1 August, 2004.
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include <strstrea.h>
void gpiberr(char *,int);                               // display GPIB error,clear device
// SEND COMMAND TO GPIB DEVICE
void gpibwrt(int device, char command[512])
{
long command_length;
command_length=strlen(command);
ibwrt(device, command, command_length);
Sleep(50);
if (ibsta & ERR){
        cout<<"device number="<<device<<endl;
        gpiberr("ibwrt Error on device no",device);
```

```
        }
}

/* ================================================
 *                  Function GPIBERR
 *  This function will notify you that a NI-488 function failed by
 *  printing an error message.  The status variable IBSTA will also be
 *  printed in hexadecimal along with the mnemonic meaning of the bit position.
 *  The status variable IBERR will be printed in decimal along with the
 *  mnemonic meaning of the decimal value.  The status variable IBCNT will
 *be printed in decimaL
 *The NI-488 function IBONL is called to disable the hardware and software.
*  The EXIT function will terminate this program.
 *
Void gpiberr(char *msg,int device)
{
   printf ("%s %i\n", msg, device);
   printf ("ibsta = &H%x  <", ibsta);
   if (ibsta & ERR  ) printf (" ERR");
   if (ibsta & TIMO ) printf (" TIMO");
   if (ibsta & END  ) printf (" END");
   if (ibsta & SRQI ) printf (" SRQI");
   if (ibsta & RQS  ) printf (" RQS");
//         if (ibsta & SPOLL) printf (" SPOLL");
//         if (ibsta & EVENT) printf (" EVENT");
   if (ibsta & CMPL ) printf (" CMPL");
   if (ibsta & LOK  ) printf (" LOK");
   if (ibsta & REM  ) printf (" REM");
         if (ibsta & CIC  ) printf (" CIC");
   if (ibsta & ATN  ) printf (" ATN");
   if (ibsta & TACS ) printf (" TACS");
   if (ibsta & LACS ) printf (" LACS");
   if (ibsta & DTAS ) printf (" DTAS");
   if (ibsta & DCAS ) printf (" DCAS");
   printf (" >\n");
   printf ("iberr = %d", iberr);
   if (iberr == EDVR) printf (" EDVR <DOS Error>\n");
   if (iberr == ECIC) printf (" ECIC <Not CIC>\n");
   if (iberr == ENOL) printf (" ENOL <No Listener>\n");
   if (iberr == EADR) printf (" EADR <Address error>\n");
   if (iberr == EARG) printf (" EARG <Invalid argument>\n");
   if (iberr == ESAC) printf (" ESAC <Not Sys Ctrlr>\n");
   if (iberr == EABO) printf (" EABO <Op. aborted>\n");
   if (iberr == ENEB) printf (" ENEB <No GPIB board>\n");
   if (iberr == EOIP) printf (" EOIP <Async I/O in prg>\n");
   if (iberr == ECAP) printf (" ECAP <No capability>\n");
```

```
   if (iberr == EFSO) printf (" EFSO <File sys. error>\n");
   if (iberr == EBUS) printf (" EBUS <Command error>\n");
   if (iberr == ESTB) printf (" ESTB <Status byte lost>\n");
   if (iberr == ESRQ) printf (" ESRQ <SRQ stuck on>\n");
   if (iberr == ETAB) printf (" ETAB <Table Overflow>\n");
   printf ("ibcnt = %d\n", ibcnt);
   printf ("\n");

/* Call the ibonl function to disable the hardware and software. */
          ibonl (device,0);
          system("PAUSE");
   exit(1);
}
// **********NPTS.CPP***********
// This function calculates the number of points to digitize in order to get
// a given number of heating periods.(acp.digtz.npts) .If  Tau is the time to
// read one point N_per is number of periods and f is the frequency then
//  NPTS = N_per * 1/f * 1/Tau. Tau is measured at TAU.CPP . Tt checks npts
// so that it doesn't become bigger then N_max, the maximum number that
// KEITHLEY can store in memory.
// It gets passed the structure ACP_ and returns tau and npts;
//Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <alloc.h>
#include <stdio.h>
#include <windows.h>
#include "decl-32.h"
#include "ACP_Ramp.h"
int NPTS(struct ACP_ Vacp,double *tau)
{
 int                    npts,
                              N_max = 5600;                    // Maximum NPTS that
KEITHLEY can store in internal memory
// for 8+1/2 digits this number is 5781 (for full read)
 float          freq = 2*Vacp.volt.freq;                  // Heating frequency is 2
times the voltage frequency.
 *tau =  Tau(Vacp);
 npts = int(Vacp.digtz.n_heat_per/freq/(*tau) + 1 );
 if ( npts>N_max || npts<=1)
 {      printf("\n\n.....Error...npts = %i; \nKeithley can't store more than 5600
readings into it's memory.", npts);
              printf("\n\nThe settings for the number of voltage periods and/or nplc ");
              printf("\nshould be changed in the acp_ramp.ini file. \n\n.....The program
will terminate.....\n\n\n");
              system("PAUSE");
              exit(1);
 }
```

```
printf("\n Npts for %i Heating Periods  = %i points.",Vacp.digtz.n_heat_per,npts);
return(npts);
}
//******TAU.CPP*******
//This function calculates Tau the time to read one point with specified
// settings from the ACP_RAMP.exe (version 2). It averages 'npts' points to find
// the average TAU . Here gets done the first communication with Keithley as
// of the ACP_RAMP program, so it gets initialized the global variable 'keith'

// It gets passed the structure ACP_ and returns tau;
// Adopted for calorrf by Saimir Barjami   on    1 August, 2004
double Tau(struct ACP_ Vacp)
{
 extern  int                    keith;                  // GP-IB address of
Keithley 2002
 extern struct ACP_             acp;
 int            i=0,                    // Dummy index
                n_filt = Vacp.digtz.n_filt,
                ch_range = 2000000,
                chann= 2,
                npts =300;      // Number of points to read to find average TAU.
 char   buffstore[100],
        trig[100],
        range[100],
        filter[100],
        func[100],
        scan[100],
        *p,                             // Dummy Variable pointer
        *keidat;                        // String where Data are buffered from
KEITH.

 Double         res,            // Variables where the stripped data from the string
        tim;                    // keidat will be stored temporarily before printing
                        // them to the file temp.dat
printf("\n\n Calculating tau -- the time to take and store into memory 1 reading.");
printf("\n For this there will be averaged %i points.",npts);
keith = ibdev(BOARD, PAD_OF_KEITH, NO_SAD,
                                        NO_TIMEOUT, EOTMODE, EOSMODE);
if (ibsta & ERR) gpiberr("Can't find Keithley online",keith);
ibclr(keith);
static char
resetctrl[]="*rst;\r\n",                                        //      Reset
controls in INIT, ARM,
resetst[]=":stat:pres;*cls;\r\n",                               // Reset status byte
statenb[]=":stat:meas:enab 512;\r\n",                   // Enable BFL(buffer full bit
```

```
msbenb[]="*sre 1;\r\n",                                          // Enable MSB
( Master Status Byte)
buffcl[]=":trac:cle;\r\n",                                       // Clear
buffer
arm[]=":arm:lay2:sour imm;\r\n",                                 // Set trigger
layer 2 to immediate
dataformat[]=":form:elem read,time;\r\n",                        // Format for downloading
over GPIB
dumpdata[]=":trac:data?;\r\n",                                   // Prepare
keithley to send data
init[]=":init\r\n",
// Initiate everything
azeroff[]=":syst:azer:stat 0\r\n";                               // Turn Autozero OFF
sprintf(func,"fres");
if (chann > 2)
 {        ch_range = 2;
          n_filt= 1;
          sprintf(func,"volt:dc"); }
  // Set channel 'chann' to measure 'func'
  sprintf(scan,"ROUTE:CLOSE (@ %i);:func '%s';\r\n",chann, func);
  // set range nplc and digits of resolution
  sprintf(range,":%s:range %i;nplc %.2f;digits 9\r\n",func,ch_range,Vacp.digtz.nplc);
  // set storage format as full and npts
  sprintf(buffstore,"trac:poin %4d;egr full;feed sens1;feed:cont next;\r\n",npts);
  //set trigger for layer 1
  sprintf(trig,"trig:COUNT %4d;\r\n",npts);
  // set number of points to filter,and  filter type. If filter <=1 do not filter
  if (n_filt <=1)  sprintf(filter,":fres:aver OFF;\r\n");
  else    sprintf(filter,":%s:aver                ON;aver:tcon                %s;:%s:aver:count
%i;\r\n",func,Vacp.digtz.filt_type,func,n_filt);
  //  Allocate memory for the variable keidat in the farheap region.
  keidat = (char *) farmalloc(4000000);
  if (!keidat)
          {
           printf("Memory allocation error for keidat - aborting\n");
           exit(1);
          }
// WRITE ALL THE COMMANDS TO KEITHLEY
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  gpibwrt(keith,statenb);
  gpibwrt(keith,msbenb);
  gpibwrt(keith,buffcl);
  gpibwrt(keith,range);
  gpibwrt(keith,scan);
  gpibwrt(keith,filter);
```

```
  gpibwrt(keith,arm);
  gpibwrt(keith,trig);
  gpibwrt(keith,buffstore);
  gpibwrt(keith,azeroff);
  gpibwrt(keith,init);                                          // Start everything
// The next line waits Keithley to assert that the buffer is full
// i.e. the corresponding bit is set, and an SRQ is required
  ibwait(keith,RQS);
  gpibwrt(keith,dataformat);
  gpibwrt(keith,dumpdata);
  ibrd(keith,keidat,4000000L);
  keidat[ibcnt]= '\0';              // Null terminate keidat at the ibcnt character.
  p = keidat;                      // p is at the begining of keidat
  sscanf(p,"%lf",&res);              // Read the first value from the String KEIDAT
  for ( ; *p != '\0'; ++p)           // Continue to read the rest of the data from KEIDAT.
  {       if (*p == ',')
                  {         i++;
                            if ( i%2!=0 ) sscanf(p+1,"%lf", &tim);
                            else sscanf(p+1,"%lf",&res);
                  }
  }

  Vacp.digtz.tau = tim/npts;                    // Calculate tau and initialize Vacp.digtz.tau
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  ibclr(keith);
// Free up the memory location allocated before.
  if (keidat)  farfree(keidat);
  printf("\n Tau = %lf s.",  Vacp.digtz.tau);
return Vacp.digtz.tau;
}
// *********  POWER.CPP      *************
//Measuring the power to the cell would be the scope of another project. We take this
power to be a constant value.
#include <math.h>
#include <alloc.h>
#include <stdlib.h>
#include <stdio.h>
#include "ACP_Ramp.h"
struct Pow_     Power(struct ACP_ acp, struct FileName fname)
{
 extern struct data     *rawdata;
 struct Pow_                                P;
 double           par[8],              // Array to hold parameters from fit
```

```
                chisq,                                          // Chi square from fit
R_std = 100.942,                    // Resistance of parallel resistor in Ohm.
                                    omega1,
                                    Vac_std,            // AC voltage amplitude across Standart
                                    Vo_std,             // DC voltage across the standart
                                    phi_std,            // Phase of voltage across standart
                                    Vac_heat,                   // Ac Voltage amplitude accros
HEATER
                                    Vo_heat,                                        //  Dc
voltage across the heater
phi_heat;           // Phase of voltage accros HEATER
 int                    i;
 FILE                   *fppar;
  omega1=2*pi*acp.volt.freq;       // Change the frequency from Hz to radians.
          P._w1 = 1;
          P.phi_w1 = 0;
          P._w2 = 1;
          P.phi_w2 = 0;
return P;
}
/// Filename     R_2_T. cpp
// This file contains the functions :
//double prtot(double res)
//double  trtot (double rval , double a[3])


/***** PRTOT.CPP **********
*  This function Converts the PRT resistance to temperature according to
*the equation R = Ro(1 + aT + bT^2) from which one finds T in degrees C
*C as a function of R and the parameters Ro_prt, a_prt,b_prt.
*  T = (1/2b)*{-a + sqrt[a^2 - 4b(1-R/Ro)]}
* PRTOT(res)  returns the temperature in degrees K,and requires as
*  argument the resistance of the PRT.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <math.h>
#include "ACP_Ramp.h"

double prtot(double res)
{
 double          x;
x = 1.0 - res/Ro_prt;
x = sqrt(a_prt*a_prt - 4.0*b_prt*x);
x = (-a_prt + x)/(2.0*b_prt);
x = 273.15 + x;
return (x);
}
```

```
//****   TRTOT.CPP   **************
// CONVERT THERMISTOR RESISTANCE TO TEMPERATURE IN KELVIN
// Converting function is 1/T = ao + a1*log(R/R0) + a2*[log(R/R0)]^2
// R0 is set at the ACP_RAMP.H header file.
// Requires as arguments the Resistance of the thermistor and the address of
// the array that contains the conversion coefficients.
double  trtot (double rval , double a[3])
{double          logr,
        tval;
        logr=log(rval/R0);
        tval=a[0]+a[1]*logr+a[2]*logr*logr;
        tval=1.0/tval;
        return(tval);
}
/* ********  Ramp_Thread.CPP  **************
*  This is the Ramping Thread. When it starts from the main thread it
* initializes LAKESHORE, and reads in from the Acp_Ramp.ini the number of zones
* and the scans for each zone. Then it starts the Ramp from the first zone to
* that the whole ramp is done so that it stops digitizing and the program ends

/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <time.h>
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <windows.h>
#include "decl-32.h"
#include "acp_ramp.h”
DWORD TempControl(LPDWORD Param)
{
        void                                                    equilb(float,float,float);
        // Set & equilibrate at the setpoint
        extern int                              lake;
        extern HANDLE                                   RampDone;
        extern struct Zone_     T_zone;
        int                                     no;
// FIND GPIB DEVICE ADDRESS  of LAKESHORE
        lake = ibdev(BOARD, PAD_OF_LAKE, NO_SAD,
                                        TIMEOUT, EOTMODE, EOSMODE);
        if (ibsta & ERR) gpiberr("Can't find LAKESHORE online",lake);
        ibclr(lake);
        gpibwrt(lake,"*cls\r\n");                                       //    Clear    the
Statusbyte register
sleep(10);
//  Start the loop that traverses all zones
```

```
for  (no=1;no<=T_zone.number;no++)
{
                    if (! strcmp(T_zone.type[no],"TIME"))Time_scan(no);
                    if (! strcmp(T_zone.type[no],"STEP"))Step_T(no);
                    if (! strcmp(T_zone.type[no],"RAMP"))Ramp_T(no);
}                           // Here is the end of all zones (For loop)
SetEvent(RampDone);               // Signal the main thread that the whole ramp is done

ibclr(lake);
ibonl(lake,0);
return 0;
}
//*******   READINI.CPP   ***************
// THIS FUNCTION READS THE RF.ini FILE NEADED TO INITIALIZE THE
// RF_RAMP EXPERIMENT.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004.
#include <dos.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <math.h>
#include "ACP_Ramp.h"
struct ACP_ Read_ini()
{
 char             ans[4];                    // Holds the answer input from user
 int              i,           // Dummy index
                  cond;        // cond==1 when some condition is satisfied
                  // cond==0 otherwise
 FILE             *fpini;
 struct ACP_              acp;
  do
  {       cond=1;
if (!(fpini=fopen("RF.ini","r")))
{  printf("\nCAN NOT OPEN RF.ini FILE. ABORTING!");
printf("\n\nMAKE SURE THE FILE RF.ini EXISTS IN THE SAME FOLDER");
printf("\nAS THE EXECUTABLE rf.EXE OR IT'S NOT CORRUPTED!");
printf("\n\nTHIS PROGRAM CAN'T RUN WITHOUT THE INITIALIZATION FILE
");printf("\nRF.INI AND WILL TERMINATE.....\n\n");
system("PAUSE");
exit(1);
}
fscanf(fpini,"%*s%*s%*s");
                  fscanf(fpini,"%*s%s", &acp.file.name);
                  fscanf(fpini,"%*s%s", &acp.file.mode);
                  fscanf(fpini,"%*s%i ",&acp.zone.number);
                  if (!(acp.zone.number < Nzones_max))
```

```
                {       printf("\nERROR IN THE NUMBER OF ZONES. ALLOWED
INPUT IS LESS THAN %i ZONES ",Nzones_max );
        printf("\nAPPLICATION WILL TERMINATE.....\n");
        system("PAUSE");
        exit(1);}
        fscanf(fpini,"%*s");
        fscanf(fpini,"%*s%*s%*s%*s");
        fscanf(fpini,"%*s");
for (i=1;i<=acp.zone.number;i++)
{  fscanf(fpini,"%s",&acp.zone.type[i]);
strupr(acp.zone.type[i]);
if  (  strcmp(acp.zone.type[i],"TIME")  &&  strcmp(acp.zone.type[i],"RAMP")&&
strcmp(acp.zone.type[i],"STEP") )
{       printf("\n\nERROR ZONETYPE IN ZONE %i. ALLOWED INPUT IS EITHER
'TIME' OR 'STEP' OR 'RAMP'",i);
printf("\nAPPLICATION WILL TERMINATE.....\n")
system("PAUSE");
exit(1);}
if (! strcmp(acp.zone.type[i],"TIME"))
{       fscanf(fpini,"%f%f",&acp.zone.start[i], &acp.zone.rate[i]);
acp.zone.end[i] = -1;}
fscanf(fpini,"%f%f%f",&acp.zone.start[i], &acp.zone.end[i], &acp.zone.rate[i]); }
fscanf(fpini,"%*s");
// Read digitizing settings.
                fscanf(fpini,"%*s%i", &acp.digtz.n_heat_per);
                fscanf(fpini,"%*s%f", &acp.digtz.nplc);
                fscanf(fpini,"%*s%i", &acp.digtz.n_filt);
                fscanf(fpini,"%*s%s", &acp.digtz.filt_type);
// Read voltage settings.
                fscanf(fpini,"%*s%f",  &acp.volt.freq);
                fscanf(fpini,"%*s%f",  &acp.volt.base_freq);
                fscanf(fpini,"%*s%lf", &acp.volt.amp);
                fscanf(fpini,"%*s%lf", &acp.volt.offset);
// Read equlibration requirements settings.
                fscanf(fpini,"%*s%i", &acp.equlbr.time);
                fscanf(fpini,"%*s%f", &acp.equlbr.deviation);
                fscanf(fpini,"%*s%s", &acp.equlbr.ctrloop);
// Read Chisq maximum and thermistor coefficients.
                fscanf(fpini,"%*s%f", &acp.digtz.chisq);
                fscanf(fpini,"%*s%*s");
                fscanf(fpini,"%*s%lf%*s%lf%*s%lf",
&acp.th.a0,&acp.th.a1,&acp.th.a2);
                fclose(fpini);
//***  Print the setup parameters read in from the Fscan.ini
                system("cls");
                printf("\nFilename = %s .",acp.file.name);
```

```
                printf(" Initial access in %s mode.",acp.file.mode);
                printf("\n#\tType\tStart (K)\tEnd (K)\t\tRate(mK/h;mK)\n");
                for (i=1;i<=acp.zone.number;i++)
                {  strupr(acp.zone.type[i]);
                if (! strcmp(acp.zone.type[i],"TIME"))
                {       acp.zone.rate[i]= fabs(acp.zone.rate[i]);
                printf("%i\t %s\t Scan @ \t %.3f for \t %.2f
Hours\n",i,acp.zone.type[i],acp.zone.start[i], acp.zone.rate[i]);
continue;
}
if (acp.zone.end[i]>= acp.zone.start[i])   acp.zone.rate[i]= fabs(acp.zone.rate[i]);
else acp.zone.rate[i]= -fabs(acp.zone.rate[i]);
printf("%i\t %s\t %.3f\t %.3f\t %.1f\n",i,acp.zone.type[i],acp.zone.start[i],
acp.zone.end[i], acp.zone.rate[i]);
}
printf("\nDIGITIZE : %i Heating Periods with %.2f NPLC.",acp.digtz.n_heat_per,
acp.digtz.nplc);
printf("\nFILTER : %i points with '%s' Type Filter.",acp.digtz.n_filt,
acp.digtz.filt_type);
                printf("\nSave Raw data file if ChiSq >= %.2f",acp.digtz.chisq);
                printf("\n\nVoltage frequency\t\t\t= %.4f mHz.",acp.volt.freq);
                printf("\n\nVoltage base frequency\t\t\t= %f Hz.",acp.volt.base_freq);
                printf("\nVoltage amplitude\t\t\t= %.4f Volts.",acp.volt.amp);
                printf("\nVoltage offset\t\t\t\t= %.4f Volts.",acp.volt.offset);
                printf("\n\nEQUILIBRATE : (for %i sec, within +/- %.0f mK )",
acp.equlbr.time,acp.equlbr.deviation);
if (*acp.equlbr.ctrloop=='a'||*acp.equlbr.ctrloop=='A')
{       printf("\nUsing the INSIDE PRT to CONTROL TEMPERATURE (Channel
%s).",acp.equlbr.ctrloop);}
else
{   printf("\nUsing the OUTSIDE PRT to CONTROL TEMPERATURE (Channel
%s).",acp.equlbr.ctrloop);}
printf("\n\nTHERMISTOR COEFFICIENTS : a0= %.3e\ta1= %.4e\ta2= %.5e",acp.th.a0,
acp.th.a1, acp.th.a2);
//***  Check that settings are OK ?!
printf("\n\nAny changes in the settings ? (Y/N)");
scanf("%s", ans);
if ((*ans=='y')||(*ans=='Y'))
{  system("cls");
        cond=0;
        printf("\n\n\nOpen the RF.ini file.");
        printf("\nDo the needed changes,and then save the file!!");
        printf("\n\nAfter you have edited the RF.ini file");
        printf("\n\nPRESS c AND HIT RETURN ( to continue) ");
        scanf("%*s");
        }
```

```
system("cls");
  } while (cond == 0);
//*** End of do loop. End of reading the parameters
 return acp;
}
/****READPRT.CPP************
*  This function reads npts readings off channel 1 of Keithley connected *
*  with the PRT at the sample.It averages them and returns to the calling*
*  function the average PRT resistance.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004.
#include <math.h>
#include <dos.h>
#include <alloc.h>
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>
#include "decl-32.h"
#include "ACP_Ramp.h"
double ReadPRT()
{
// ***********BEGIN INTERNAL DECLARATIONS ************
int       npts= 10,
          stb_value,
          digits =9;                        // Resolution for readings.
extern  int keith;                          // GP-IB address of Keithley 2002.
char      buffstore[100],     // Strings that hold specific commands for KEITH.
                  trig[100],
                  fres[100],
                  SP_r[10],
                  *p;                               // Dummy Variable pointer.
  char            *keidat;      // String where Data are buffered from KEITH.
  double          res,          // Variable holding data from the buffered 'keidat'
                  aveprt,                   // Average value of PRT
                  sum;                              // Sum of the PRT readings.
  static char resetctrl[]="*rst\r\n",                                   //  Reset  controls  in
INIT, ARM,//  and put triger in IDLE state
resetst[]=":stat:pres;*cls\r\n",          // Reset status byte
statenb[]=":stat:meas:enab 512;\r\n",       // Enable BFL(Buffer Full bit
msbenb[]="*sre 1\r\n",                                                  //                  Enable
MSB(Master Status Byte)
 buffcl[]=":trac:cle\r\n",                                                  // Clear buffer
// Take (Npts) readings,after the triger is set IMMEDIATELY.
 arm[]="arm:lay2:sour imm;\r\n",
// Set channel 1 to measure 4W resistance;
 scan[]="ROUTE:CLOSE (@ 1);:func 'fres';\r\n",
// Store data into buffer;speciffies buffer size(3 points) and full format
```

```
dataformat[]="form:elem read;\r\n",
// Set Keithley to dump the data of the buffer, when the IBRD command is sent
dumpdata[]="trac:data?;\r\n",
// Set autozero ON for the Prt.This allows recalibration with internal standarts.
azeron[]=":syst:azer:stat 1;:syst:azer:type norm;\r\n",
// Initiate everything
 init[]=":init\r\n";
sprintf(fres,":fres:range 20000;nplc 10;:fres:digits %d;\r\n",digits);
  sprintf(buffstore,"trac:poin %i;egr full;feed sens1;feed:cont next;\r\n",npts);
  sprintf(trig,"trig:COUNT %4d;\r\n",npts);
//************END OF DECLARATIONS ******************************
//  Allocate memory for the variable keidat in the farheap region.
  keidat = (char *) farmalloc(40000);
  if ((!keidat) ) {
            printf("Memory allocation error - aborting\n");
            exit(1);
   }
  ibclr(keith);
// WRITE ALL THE COMMANDS TO KEITHLEY
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  gpibwrt(keith,statenb);
  gpibwrt(keith,msbenb);
  gpibwrt(keith,buffcl);
  gpibwrt(keith,fres);
  gpibwrt(keith,scan);
  gpibwrt(keith,arm);
  gpibwrt(keith,trig);
  gpibwrt(keith,buffstore);
  gpibwrt(keith,azeron);
/* Start everything  */
  gpibwrt(keith,init);
// The next line waits Keithley to assert that the buffer is full
// i.e. the corresponding bit is set, and an SRQ is required
//  ibwait(keith,RQS);
// Actually we are polling the status byte.
  sleep(12);                        // Sleep while Keithley is reading the PRT
  do
  {         gpibwrt(keith,"*STB?\r\n");
            ibrd(keith,keidat,9L);
            keidat[ibcnt]= '\0';
            stb_value = atoi(keidat);
           } while (stb_value != 65);
           ibrsp ( keith, SP_r );              // serial poll keithley to release SRQ
// Send the dataformat, and read from buffer into keidat.
  gpibwrt(keith,dataformat);
```

```
  gpibwrt(keith,dumpdata);
  sleep(0.2);
  ibrd(keith,keidat,40000L);
// Null terminate keidat at the ibcnt character.
  keidat[ibcnt]= '\0';
//  Read from keidat the readings and write them at the file temp.dat
//  Time and resistance.
//  printf("\n Stripping keidat and writting at temp.dat.");
  p = keidat;
// Read the first value from the String KEIDAT
  sscanf(p,"%lf",&res);
  sum = res;
// Continue to read the rest of the data from KEIDAT.
  for ( ; *p != '\0'; ++p)
  {
          if (*p == ',')
          {       sscanf(p+1,"%lf", &res);
                  sum=sum+res;
          }
  }
  aveprt= sum/npts;
  gpibwrt(keith,resetctrl);
  gpibwrt(keith,resetst);
  ibclr(keith);
// Free up the memory location allocated before.
  if (keidat)  farfree(keidat);
  return (aveprt);
// END OF FUNCTION.
}
//********SAVE.CPP ****************
// Contains two functions.
// Save_data saves data and fit parameters at the .dat,.par file respectively .
// Save_Raw_Data saves the raw data at the file with a .raw extension and name +
// point number name. It saves the parameters of the fit at the same file.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <time.h>
#include <stdio.h>
#include "ACP_Ramp.h"
void Save_Data( double data[10], double par[NFUNC],struct FileName name,int ptno)
{
          FILE            *fpdat,
                                  *fppar;
          char            time[30];
          int             i;
          _strtime(time);
          fpdat=fopen(name.datafile,"a");
```

```
        fprintf(fpdat,"%i \t%9.5f \t%9.5f \t%e \t%e\t%e\t%e \t%e\t%e\t%e\t%7f
\t%s\n",ptno,data[0],data[1],data[2],data[3],data[4],data[5],data[6],data[7],data[8],data[9]
,time);
        fclose(fpdat);
        fppar = fopen(name.parfile, "a");
        fprintf(fppar,"%i",ptno);
        fprintf(fppar,"\t%9.5f",data[0]);
        for (i=0;i< NFUNC; i++)
        {
                fprintf(fppar,"\t%e ",par[i]);
        }
        fprintf(fppar,"\t%lf",data[9]);
        fprintf(fppar,"\t%e\n",data[10]);
        fclose(fppar);
        return;
}
void Save_Raw_Data (struct data *rawdata, double *par,int ptno, double chisq, struct
ACP_ acp)
{       int                                                     k, i;
        FILE                                                    *fpraw;
        struct data                             *dp;
        char    name[100];
        sprintf(name,"%s_%i.raw",acp.file.name,ptno);
        fpraw=fopen(name,"w");
        for (k=0;k< NFUNC; k++)
        {fprintf(fpraw,"%lf\t ",par[k]);}
        fprintf(fpraw,"%lf\n",chisq);
        dp= rawdata;
        for (i=0; i<acp.digtz.npts; i++,dp++)
        {fprintf(fpraw,"%9.6f\t%9.2f\n",dp->time,dp->value);       }
        fclose(fpraw);
        return;
}
//***** THRED_EX.CPP*********
// This file contains the declarations of the Handles and signals needed for
// the multithreaded RF_RAMP.EXE 2.0  . All are global variables.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <stdio.h>
#include <windows.h>
void Ini_Thread_Var()
{
 extern HANDLE                  ZoneDone;                                       //
Signals when the n-th zone ramp is done
 extern HANDLE                  RampDone;                                       //
Signals when the whole ramp is done
```

```
 extern HANDLE              StartDigitize;                              // Signals the
main thread when to start digitizing.
 extern HANDLE              T_Eqlbr_for_Step;          // For step type zone signals that
equilibrium is achieved
 extern HANDLE              Done_Digitizing;           // Signals the stepping thread to
move at the next T .
// Create the signaling event when the ramp is done.
 RampDone = CreateEvent( NULL,                                  // Default security
attributes
                                                                FALSE,
                // Automatically reset after wait
                                                                FALSE,
                // Initial State is *** Ramp NOT Done ***
                                                                NULL
                // No name for the event
                                                        );
// Create the signaling event to start digitizing after the equilibrium is reached.
 StartDigitize = CreateEvent( NULL,                     // Default security attributes
                              FALSE,                    // Automatically reset after
wait
                              FALSE,                    // Initial State is *** NOT
Done ***
                              NULL                      // No name for the event
                                                                );
// Create the signaling event to start digitizing after the equilibrium is reached.
 ZoneDone = CreateEvent( NULL,                                  // Default security
attributes
                              FALSE,                                    //
Automatically reset after wait
                              FALSE,                                    // Initial State
is *** Zone NOT Done ***
                               NULL                                     // No name for
the event
);
 // Create the signaling event to signal the main thread that the equilibrium is reached.
 T_Eqlbr_for_Step = CreateEvent( NULL,                  // Default security attributes
        FALSE,              // Automatically reset after wait
        FALSE,              // Initial State is *** Equilibrium not reached ***
        NULL                // No name for the event
                                                                )
Done_Digitizing = CreateEvent( NULL,                    // Default security attributes
        FALSE,              // Automatically reset after wait
        FALSE,              // Initial State is ** not done with digitizing ***
        NULL                // No name for the event
                                );
        return;
```

```
}
// *******   WRT_HEAD.CPP    *************
// CONTAINS TWO FUNCTIONS.
// Write_Header writes the header at both.dat and .par files
// Write_SubHeader wites the subheader at the end of each zone at the .par file.
/* Adopted for calorrf by Saimir Barjami   on    1 August, 2004
#include <time.h>
#include <stdlib.h>
#include <stdio.h>
#include "ACP_Ramp.h"
void Write_Header(struct FileName name, struct ACP_ acp)
{
 char    time[30],              // Holds the time returned from the CPU
         date[30];         // Holds the date returned from the CPU
 int     i;
 FILE   *fpdat,
         *fppar;
// Write headers of the files .dat and .par if the file is opened in W mode
  if (!(fpdat=fopen(name.datafile,acp.file.mode)))
                  {   printf("\nCAN NOT OPEN %s FILE. ABORTING!",name.datafile);
                   exit(1);
                  }
fprintf(fpdat,"\n**********************************************************
*********");
 fprintf(fpdat,"\n\t\t**** Calor A. AC Calorimetry. ACPSF   ALL Program. ****");
fprintf(fpdat,"\n*********************************************");
 fprintf(fpdat,"\n\nFILE NAME = %s ;",name.datafile);
//*** Read date and time from the computer clock and write it in the file.
  _strtime(time);
  _strdate(date);
  fprintf(fpdat,"\nFile opened on %s at %s. ",date,time);
  fprintf(fpdat,"\n\nNpt                                                                    \tTPRT
\t\tT_THERM\tR_AVG\t\t\tTac_w\t\t\tCp_w\t\t\tPhi_w\t\t\tTac_2w\t\tCp_2w\t\t\tPhi_2w
\t\tChisq\t\tTIME\n");
fclose(fpdat);
if (!(fppar=fopen(name.parfile,acp.file.mode)))
{   printf("\nCAN NOT OPEN %s FILE. ABORTING!",name.parfile);
exit(1);
} fprintf(fppar,"\n***************************");
 fprintf(fppar,"\n\t\t**** Calor A. AC Calorimetry. ACPSF ALL Program. ****");
fprintf(fppar,"\n*******************************");
fprintf(fppar,"\n\nFILE NAME = %s ;",name.parfile);
//*** Read date and time from the computer clock and write it in the file.
 _strtime(time);
 _strdate(date);
```

```
 fprintf(fppar,"\nFile opened on %s at %s. Calculated duration : %.1f hours",date,time,acp.zone.tot_t);
 //*** Write the scans settings in the file.
  fprintf(fppar,"\n\n\tThese are the settings for the zones.");
  fprintf(fppar,"\nZone # \tZone Type\t\tStart T\t\tEnd T\t\t\tRate(mK/h;mK;H)\n");
  fprintf(fppar,"-------------------------------------------------------------------------------\n");
  for (i=1;i<=acp.zone.number;i++)
 {
fprintf(fppar,"%i\t\t%s\t\t\t%6.3f\t\t%6.3f\t\t%.2f\n",i,acp.zone.type[i],acp.zone.start[i],acp.zone.end[i],acp.zone.rate[i]);
}  fprintf(fppar,"\n*****************************");
fprintf(fppar,"\nDigitizing %i Heat Periods,@ %.2f nplc integration",acp.digtz.n_heat_per,acp.digtz.nplc);
fprintf(fppar,"\nFiltering : %i points with %s Type Filter.",acp.digtz.n_filt, acp.digtz.filt_type);
fprintf(fppar,"\nEquilibrate for %i sec, within +/- %.1f mK", acp.equlbr.time,acp.equlbr.deviation);
fprintf(fppar,"\nVoltage frequency = %.3lf mHz",1000*acp.volt.freq);
fprintf(fppar,"\nVoltage amplitude = %.5lf V",acp.volt.amp);
fprintf(fppar,"\nVoltage offset = %.5lf V",acp.volt.offset);
if (*acp.equlbr.ctrloop=='a'||*acp.equlbr.ctrloop=='A')
{       fprintf(fppar,"\nUsing the INSIDE PRT for temperature control (Channel %s).",acp.equlbr.ctrloop);}
else
{     fprintf(fppar,"\nUsing the OUTSIDE PRT for temperature control (Channel %s).",acp.equlbr.ctrloop);}
  fprintf(fppar,"\nThermistor coeficients are:");
  fprintf(fppar,"\n a[0]= %e\n a[1]= %e\n a[2]= %e",acp.th.a0, acp.th.a1, acp.th.a2);
  fclose(fppar);
}
void Write_SubHeader(struct FileName name, int i)
{
FILE              *fppar;
fppar=fopen(name.parfile,"a");
fprintf(fppar,"\n**********  ZONE # %i .  **********",i);        // Print on file the zone number.
fprintf(fppar,"\nNpt\tT_prt\t\tSIN_w\t\t\tCOS_w\t\t\tSIN_2w\t\tCOS_2w\t\tT^1\t\t\tT^2\t\t\tT^3\t\t\tCHISQ\t\tDelta Tprt\n");
fclose(fppar);
return;
}
```