Transient Thermal Recovery of a Pulsed BSCCO-2223 Superconducting Magnet

by

Karl W. Kowallis

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Feb 2000

Author ....
Department of Mechanical Engineering
Jan 14, 2000

Certified by.. ....
Yukikazu Iwasa
Research Professor, Francis Bitter Magnet Laboratory, and Senior
Lecturer, Department of Mechanical Engineering, MIT
Thesis Supervisor

Accepted by ....
Ain A. Sonin
Chairman, Departmentmental Graduate Committee

Transient Thermal Recovery of a Pulsed BSCCO-2223 Superconducting Magnet

by

Karl W. Kowallis

Submitted to the Department of Mechanical Engineering
on Jan 14, 2000, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

High-temperature superconducting electric devices such as fault-current limiters, transformers, motors and power lines require protection from fault-mode overcurrent pulses that cause normal heating and conductor damage. Experimental observation of a hight-temperature superconducting magnet subjected to controlled overcurrent pulses and a quantitative numerical representation are the objectives of this study.

A square shaped current pulse on a BSCCO-2223 superconducting magnet can exceed the critical current of the conductor and drive it normal. Lowering the current magnitude leaves 3 possible conditions for the magnet: superconducting, normal but recovering, and continuing to quench. The test coil has 2 concentric sections, the inner wound on a copper coilform and the outer wound over the inner section. Each section is dry wound with silver-sheathed Bi-2223 composite tape, with each conductor layer insulated with three layers of Kapton tape. The current carrying capacity of the inner tape is greater than that of the outer tape.

The test magnet, coupled to a cryocooler, is housed in a cryostat and a data acquisition system recorded voltage measurements. The experiment demonstrated that the test magnet, initially at a given operating temperature, did display the anticipated quench conditions when subjected to overcurrent pulses. A numerical simulation (Lax method) based on a Cartesian approximation of the heat equation predicts the voltage on the test magnet at currents near critical, and joule heating profiles match experimental observation, but lack explicit agreement.

Discrepancies between experiment and simulation are attributed to significant contact resistance and unknown thermal interaction between the copper coilform and the inner coil and also between the two sections.

Thesis Supervisor: Yukikazu Iwasa
Title: Research Professor, Francis Bitter Magnet Laboratory, and Senior Lecturer, Department of Mechanical Engineering, MIT

Acknowledgments

I owe a great deal to those who have furthered my education throughout my life. I could write volumes trying to name all of them. Thank you. I am especially grateful to a wonderful and loving family. You have always supported me in both good times and bad. I am also appreciative of the friends who have associated with me during my time here at MIT, you have changed my life for the better.

Many individuals deserve thanks for the creation of this thesis. Thanks to Dr Iwasa for his insight and instruction. Because of his knowledge on this subject I was forced to learn, because I couldn't pretend to know what I was talking about. Thanks to Haigun Lee for frequent smiles and compliments, they were always appreciated. Without Ben Haid I would never have survived my experimentation and simulation. Thanks for tutoring me in the lab and providing so much foundation on which to begin my own work. You were always available with answers when I was stuck. Your moral support and prior experience helped me believe I could finish this project.

To those not mentioned by name, thank you. My prayers are for you, and with gratitude because of you. Sometimes it helps to think of things in their proper perspective and then realize what is really important.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Superconductivity

Superconductivity is the point reached by some materials at low temperature where they loose their measurable electrical resistivity. Kamerlingh Onnes, a Dutch physicist, discovered the superconductivity of mercury in 1911 when he cooled it to 4.2 K, the boiling point of helium at atmospheric pressure. Several other materials were discovered by Kammerlingh Onnes and his successors, none of which had commercial application due to their low current densities. In the 1950's a new type of superconducting material was developed, called Type II, supporting larger critical current densities. For several decades the variety and properties of these new superconductors were analyzed and great efforts were made to increase the critical temperatures.

1.2 HTS

Beginning in 1986 with the discovery of an oxide of Ba-La-Cu-O with a critical temperature of 35 K, a race to discover materials with even greater critical temperatures was on. This 35 K mark was more than 10 K above the previous high temperature record holder and opened the way for a multitude of new materials. Within a few years critical temperatures exceeded 100 K and materials with critical temperatures above that of liquid Nitrogen, 77 K, became available. These "high temperature" superconductors that can be cooled by nitrogen rather than helium are more efficient, from a cryogenics standpoint, and are often cooled "dry" using cryocoolers, a low temperature refrigerator.

The drawback of these new ceramic high-temperature conductors is their material properties. They are brittle, hard to form, and subject to reduced electrical capacity resulting from strains and variations in production. All of this leads to high expense and low flexibility.

1.3 Thermal properties

Superconductors require low temperatures to maintain their superconducting properties. If that superconducting state is lost due to mechanical disturbances, localized heat introduction, or simply variations in material properties within the conductor, the incurred resistance becomes an additional source of heat, spreading this non-superconducting region. This "normal" region may develop a run-away instability, remain a fixed size due to the conduction of heat away from the normal area, or may shrink if the thermal conduction is greater than the heat being generated. Low temperature superconductors have a fairly well defined temperature line between the superconducting and normal states. The "Normal Zone Propogation" (NZP) can be estimated analytically and measured experimentally.

In HTS, the superconductor is embedded in a matrix of conductive metal, often silver, that acts as a set of parallel paths for current. As current exceeds the capacity of the superconductor, it flows through the resistive metal generating heat. There is a wide temperature region in which "current sharing" occurs. HTS also are operated in the dry, without a liquid cryogen, and behave adiabatically during "normal events." Benjamin Haid has modeled the NZP in a BSCCO-2223 coil using experimental data correlated to numerical simulation.[4] The larger range of potential operating temperatures for these high temperature superconducors provides greater opportunity for thermal recovery from a localized quench. By eliminating liquid cryogen cooling channels higher current densities are also possible, at the overal expense of greater opportunity for conductor damage in the event of un-checked normal zone growth.

1.4 Experimental purpose

Designing traditional electric devices using superconductors poses a challenge in anticipating fault conditions. Devices such as transformers, motors, and power lines are exposed to occasional bursts of unexpected current. As this current goes beyond the critical capacity of the superconductor, the conductor is driven normal. Depending on the cooling design and the safety factors in the design, this overcurrent will result in a universal quench involving conductor damage. If this initial pulse is stepped down there are three possibilities, based on the cooling capacity of the cryogenics and the temperature to which the conductor was raised in its normal phase.

- A runaway quench. This will damage the conductor if the current load is not removed.
- Non-quench. The reduced current following the pulse may be within the conductors superconducting range at the operating condition, allowing diffusive cooling to the cryogenic heat sink.
- A thermal recovery. Joule dissipation removal lowers the conductor temperature.
- A psudo stability. Heat removal approximately matches the heat generation and a psudo-steady state is observed. In reality this state eventually leads to a quench or recovery.

The purpose of this project has been to experimentally observe each of these cases and to create a numerical model that approximates the observed behavior. It is hoped that this research will verify the assumptions regarding the cooling of superconductors in general, and superconducting magnets in particular. It will also provide a prediction tool for fault tolerance in commercial applications.

Chapter 2

Experimental Setup

The experimental results were obtained from a 240-turn BSCCO-2223 coil constructed by Sumitomo Electric. Voltage taps were attached at the extremities of the two coils, which were in series, and voltage taps were attached to several turns on the outer layer.

2.1 Overall operation

The Sumitomo coil is composed of two separate, concentric coils, attached in series and wrapped around a copper coilform. The copper leads from the current port are attached to HTS leads mounted on a copper extension rod from the second stage of a Daikin cryocooler, model U104CWZ. Figure 2-1 shows the experimental setup with the cold components housed in a stainless steel cryostat.

Both the sample space and the vacuum space are evacuated with a diffusion pump to 10^{-5} torr to minimize convective heat transfer. Twenty five layers of aluminized mylar super-insulation are wrapped around the magnet and copper extension to reduce radiation input to the system. The copper extension rod couples the second stage of the cryocooler and the magnet with less than a 1K temperature difference. The temperature of the coilform is measured by a calibrated GaAlAs thermodiode. Type E thermocouples are anchored to the lower level of the copper coilform, at the level of the diode, and attached to the surface of the magnet.

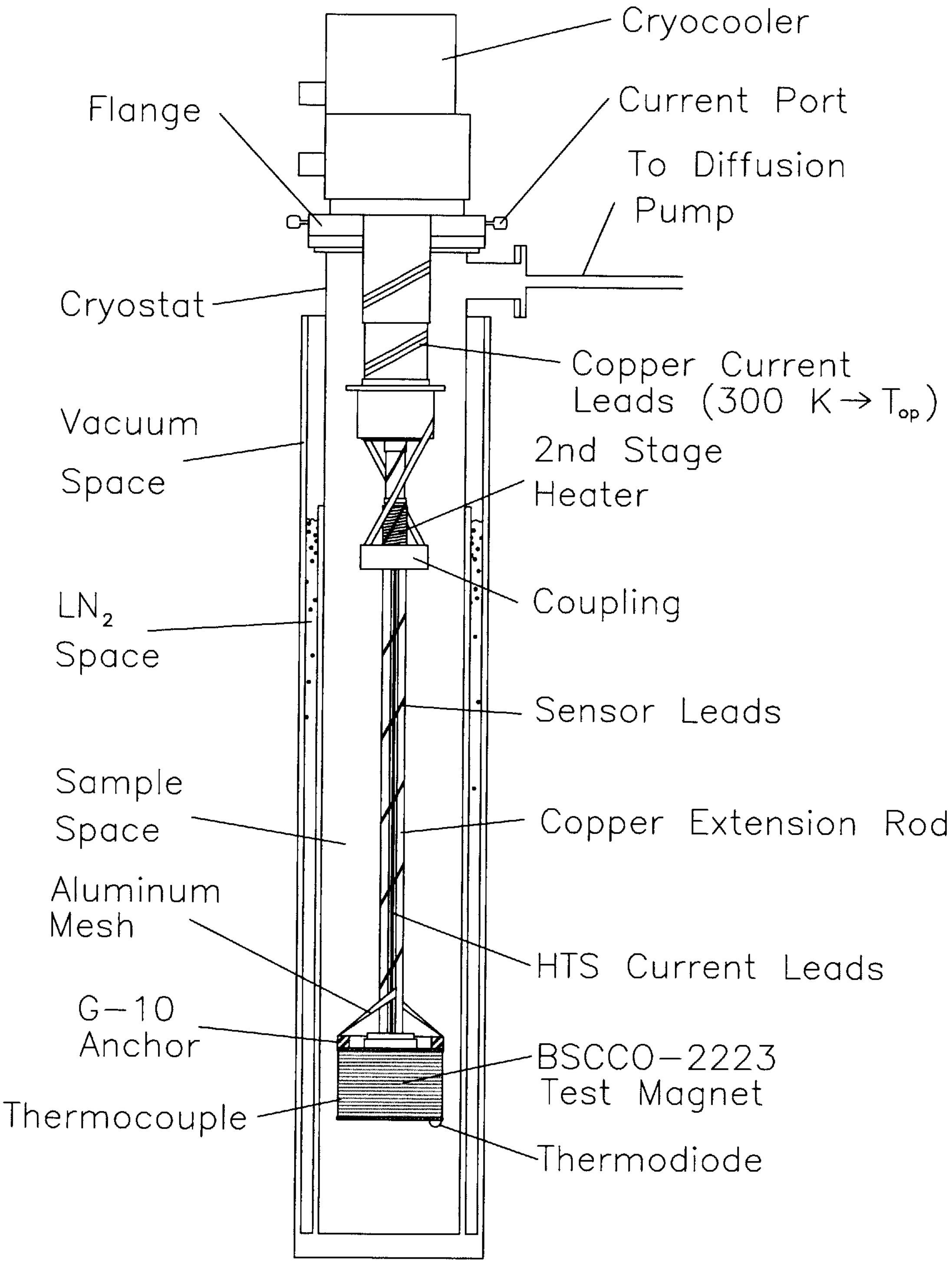

Figure 2-1: Experimental Setup of test coil, dewar and cryocooler

Current is supplied by two Hewlett Packard power supplies operating in parallel. The supply leads are clamped to the current lead outlets, which extend from the flange that joins the cryocooler and cryostat. Copper cable connects the current port to HTS leads on the copper extension. The copper cable is also anchored by a copper post to the first stage of the cryocooler to reduce the heat input along the leads. The HTS leads, of the same BSCCO-2223 tape as the magnet is made of, travel the length of the copper extension and are connected to the magnet leads by aluminum mesh.

The power supply is turned on and off by hand for each experiment and the change from pulse to operating current is done by an adjustable switch using resistive programming to control the power supply. All pairs of voltage taps to be measured as well as leads from the thermocouples travel through a 32 pin connector in the cryocooler flange and are attached to BNC cables into a D/A converter. Measurements are done on a digital computer with Labview software and coil voltages, diode voltage and power supply shunt voltage are all measured in real time using Keithly voltmeters.

2.2 Thermal systems

2.2.1 Cryostat / Cryocooler Assembly

The cryostat consists of two sealed spaces, the vacuum space, which surrounds a liquid nitrogen space, and the sample space. When the vacuum space and sample space are evacuated, heat transfer between the walls and the test magnet are minimized. A vacuum valve allows the diffusion pump to be disconnected from the vacuum space while a vacuum is maintained. A vacuum flange at the top of the sample space permits continuous pumping and the pressure is measured using a discharge gauge mounted upstream of the diffusion pump.

The vacuum space is constructed with stainless steel sheets and the nitrogen space is open to the atmosphere. The stainless steel inner wall faces the aluminized copper wall of the sample space. Filling the nitrogen space with liquid nitrogen reduces the radiation input. Without the radiation reduction the temperature difference between the magnet winding and the coilform may be as high as 30K.

A flange at the top of the sample space is constructed from 3/16 inch stainless steel and the cryocooler and cryostat flanges are joined by bolts to an intermediate 1/2 inch stainless steel flange. Silicon greased Parker O-rings sit in canals on top of each flange for a tight vacuum seal.

2.2.2 Copper Extension Rod

The copper extension rod (see Figure 2-1 on page 14) serves to thermally connect the test coil to the cryocooler second stage while keeping the cryocooler away from the magnetic fields produced by a high field magnet. (The original setup was designed to expose the test coil to a magnetic field as high as 8 T, generated by an external electromagnet.) The cryocooler contains a permanent magnet stepper motor that drives a piston for conducting the helium flow. Too high a magnetic field will cause the motor to saturate.

The rod is 50 cm long and has a square cross section with 28 mm sides. Apezion grease is applied at both joints to improve the thermal connection. Two pairs of BSCCO-2223 tape strips are epoxied to the sides of the rod as current leads. Below the epoxy is a Kapton barrier to electrically insulate the leads. The wires attached to the thermocouples and voltage taps are wrapped around the rod several times to reduce their heat input into the system and prevent temperature variation at the tap connection.

2.3 Electrical Systems

2.3.1 Current Source and Copper Leads

The electrical setup is diagrammed in Figure 2-2. Current is supplied by two Hewlett Packard 6260B 100A, 10V power supplies operating in parallel. A 200 $\mu\Omega$ shunt placed in series with the current output allows the current to be measured based on

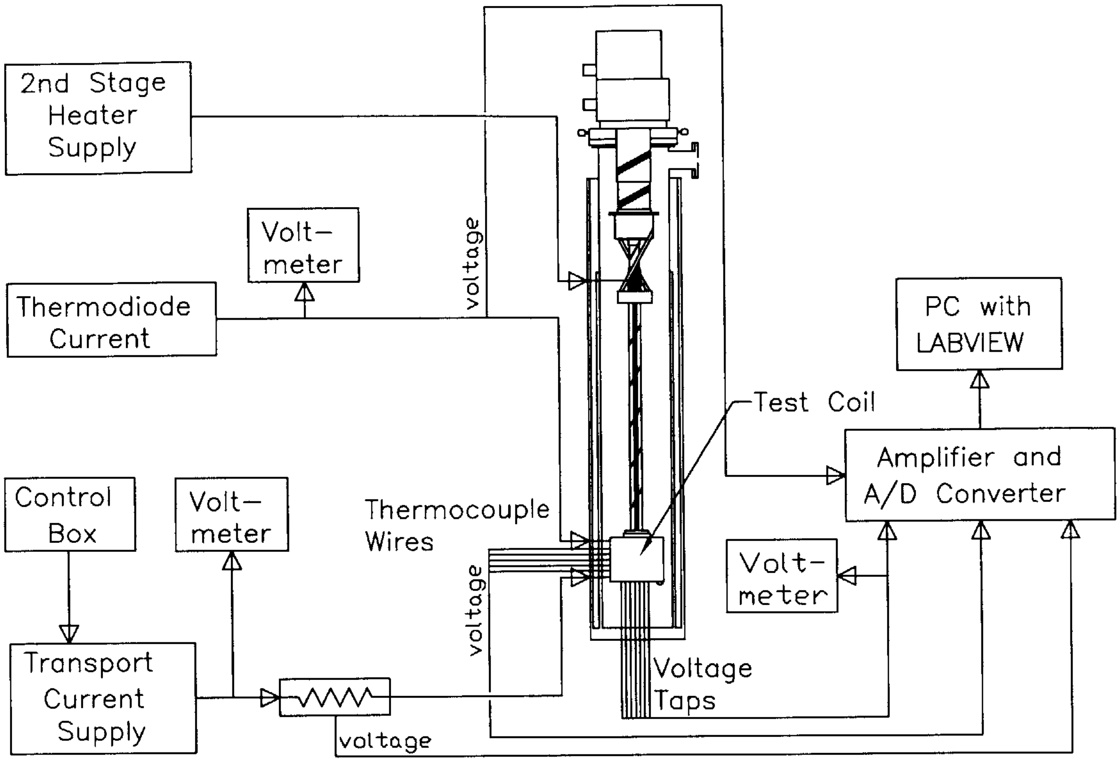

Figure 2-2: Electrical schematic of test setup

the voltage drop across the shunt. Current control is by resistance programming to the terminal strips on the back and is attached to a control box with two independent potentiometers and a dial for switching between the two.

The supply leads are clamped to the current ports which extend from the flange between the cryocooler and the cryostat. The current ports are 1/4 inch diameter copper pins that are electrically isolated from both the flange, cryocooler and cryostat and sealed with epoxy. The copper current pins are attached internally to copper cable that is thermally anchored to the first stage of the cryocooler and attached to the HTS leads just below the second stage on the copper extension.

2.3.2 HTS Current Leads

Two pairs of silver-sheathed BSCCO-2223 tape strips carry current from the copper cable to the magnet's aluminum mesh leads. The use of HTS leads is to reduce heat input and the leads are electrically isolated from the copper extension with Kapton

Table 2.1: Test Coil Parameters

	Inner Section	Outer Section
Inside Diameter [mm]	100.0	105.8
Outside Diameter [mm]	104.6	108.1
Wind Height [mm]	76.0	76.0
Total Turns	160	80
Layers	8	4
Turns/ Layer	20	20
Conductor Length [m]	52	27.6
Self Inductance [mH]	1.43	2.37
Mutual Inductance [mH]	1.72	1.72
Center Field @100A [T]	0.11	0.13
Peak Field @100A [T]	0.13	0.17
λJ @ 100A [A/m^2]	9.21×10^7	$8.98\ \times10^7$

Table 2.2: Conductor parameters

	Inner Section	Outer Section
Conductor Width [mm]	3.5	3.5
Conductor Thickness [mm]	0.23	0.23
I_c (77 K, 0 T) [A]*	23.5	19.9
Silver Fraction	0.70	0.70
*$E_c = 0.1\,\mu V/\mathrm{cm}$		

and epoxied in place. The conductor is manufactured by Sumitomo and is rated 42 A at 77 K.

2.3.3 Test Coil Construction

Detailed drawings of the test coil, manufactured by Sumitomo Electric, and its heat lead connection are in Figure 2-3. The coil consists of two concentric sections of equal height and the same wind direction. The outer and inner conductors have different critical currents, due to conductor quality, but the same cross section. The coils are dry wound and tension is maintained by a double stick pad at the end of each layer. The conductor is wrapped once in 25 μm thick Kapton insulation, as depicted in Figure 2-4. There are effectively three layers of Kapton between each layer of conductor. Dimensions of the test magnet are listed in Table 2.1, and conductor specifications in Table 2.2.

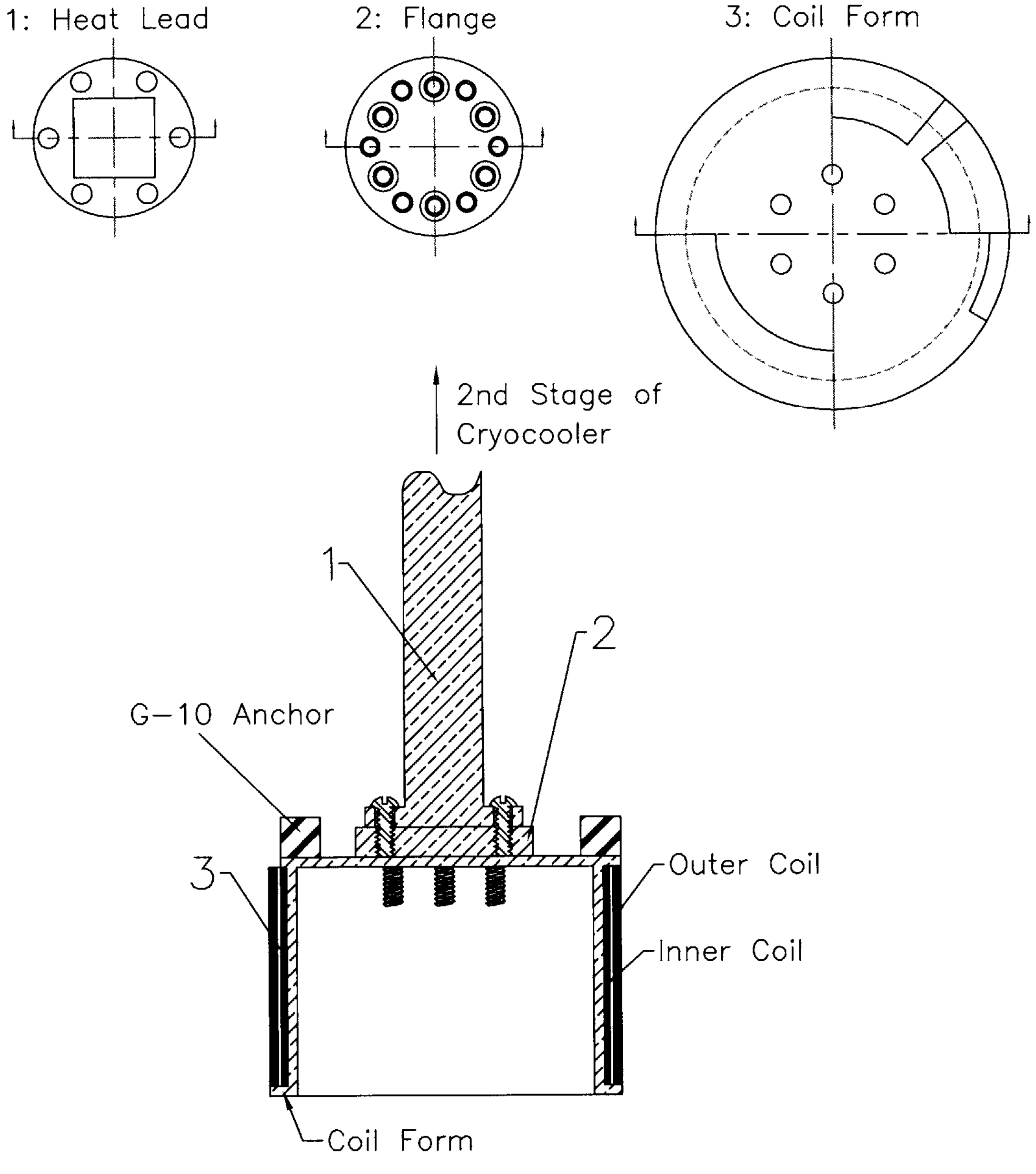

Figure 2-3: Detailed drawing of the test coil coupled to the copper extension rod heat lead. Parts 1, 2, and 3 are of copper so that the temperature at part 3 is within a fraction of a Kelvin from that of the second stage.

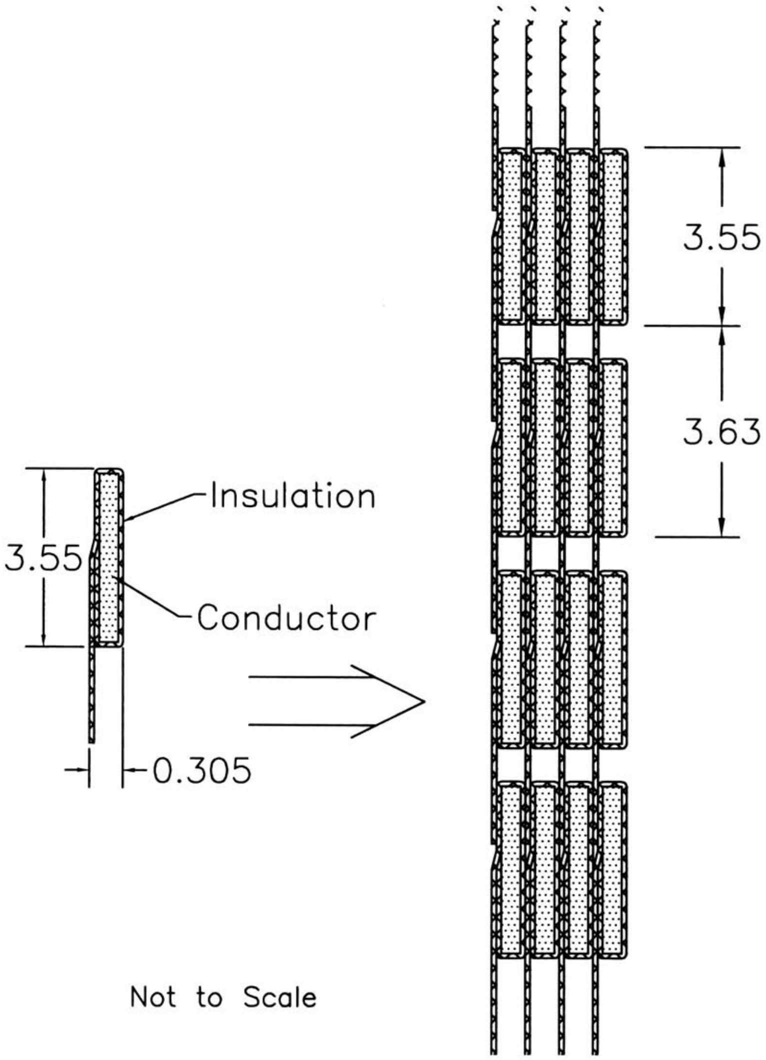

Figure 2-4: Insulation detail. Dimensions are in mm.

Soldesign, a solenoid design computer program, is used to calculate the field profile and inductances. For a transport current of 100 A, the test coil generates a centerline field of 0.24 T and a peak field of 0.30 T. [12]

The coilform and the extension are of copper, facilitating high thermal diffusion and maintaining fairly uniform temperatures along the supporting structures. The ends of the conductor attach to G-10 anchors, where they are soldered to aluminum mesh connected to the HTS leads.

2.3.4 Voltage Taps

Each voltage tap is a 3-mm wide strip of copper shim, slipped beneath the Kapton insulation and secured and electrically contacted with BIPAX epoxy. The silver BIPAX epoxy is electrically conductive and can be applied without any heating to prevent conductor or insulation damage. Wires from the 32 pin connector are wrapped around the copper extension and soldered to the copper taps.

2.3.5 Temperature Control

Surrounding the second stage of the cryocooler are 20 turns of manganin wire, wrapped in a layer of Apiezon-N thermal grease. The resistive power generated controls the baseline temperature achievable by the cryocooler. A supply of 2.1 W achieves 20 K, 6.1 W for 30 K, and 7.7 W for 40 K.

2.3.6 Temperature Measurement

The temperature of the copper coilform is measured with a calibrated Lakeshore Cryogenics GaAlAs thermodiode. The diode sits in a 2.0 mm diameter by 1.5 mm deep hole at the bottom edge of the coilform. The recess is filled with Apiezon-N thermal grease and the diode is taped in place. Wire leads are attached to the diode and the single set of leads are used for both the 10 μA current source, supplied by Lakeshore Cryogenics, and the voltage measurement.

Type E thermocouples are isolated from the coilform by a layer of Kapton and then thermally anchored around the bottom edge using thermal grease and compression taping. The end of the thermocouples are coated with a thin layer of Apiezon-N grease and slipped under the outer layer of Kapton on the magnet. The assumption is made that the anchored ends are at the same temperature as the diode measurement and calibration for the diode is used to generate a reference voltage that is added to the thermocouple voltages. This derived voltage is then turned back into a temperature using an interpolating polynomial. All calculations are done in MATLAB following the data collection.

2.3.7 Instrumentation

All voltage measurements are sampled using LabView software running on a Macintosh digital computer. The computer receives inputs from a National Instruments A/D converter with built in isolation amplifiers. Amplification ranged from 1-1000, with the smaller for the coil voltages themselves and the largest for the thermocouples, which measure all voltages in mV. Filters were set at 10 kHz and a 50 Hz sample rate was used. Real time voltage measurements were observed using Keithley 175, 191, and 197 voltmeters.

2.4 Procedure

Once the cryocooler is running and the coilform has reached a steady state, liquid N_2 is placed in the cold space to decrease the radiation input and the temperature control heater is set to an appropriate level. To reach a uniform temperature within 3K of the desired coilform temperature may take 12 hours, with minor current adjustments to the heater needed to achieve the desired temperature. Even after the coilform temperature has reached the desired temperature the coil temperature may vary from it by 2 - 6 K. The coil is generally the warmest just below center on the outer layer and coolest on the upper turn of the innermost layer, which is next to the coilform. Once the conditions have reached the desired values the following procedure is used to collect data.

1. The filename for the data collection is set in LabView and the chosen sample rate, filename and conditions are written down in a lab book.

2. Potentiometer levels are checked on the power supply to confirm the pulse and operating current levels.

3. LabView data collection is initiated.

4. The power supply is turned on and real-time voltmeters observed.

5. After the pulse length, typically 3 seconds, the current is stepped to the operating value.

6. After approximately 10 seconds at the operating current the power supply is shut off.

7. Data files from LabView are saved and the acquired data is observed.

8. Depending on the level of quenching, the next trial may begin in as soon as 10 minutes or as long as 4 hours.

Throughout the process the real-time voltmeters are observed to be prepared for a quench. If a quench is initiated it is carefully observed as the voltages rise. If the voltage increases too drastically the trial may be manually cut short by turning off the power. Failure to shut off the current can lead to a burnout in the magnet, which requires that the magnet be removed from the experimental set up and either the burned section repaired, or the entire layer containing the damage removed.

A summary of the trials and the operating parameters is found in Table 2.3. I_p is the pulse current and I_0 is the operating current after the pulse.

Table 2.3: Experimental Trials Summary

Trial	T_0	I_p	I_0	V_{max}	V_{max}	operating current
	[K]	[A]	[A]	inner [V]	outer [V]	result
1	20	123.9	36.0	0	0.016	superconducting
2	20	135.0	51.9	0	0.073	superconducting
3	19	159.2	29.4	0	0.352	steady voltage
4	39	120.2	27.6	0.01	0.049	superconducting
5	39	128.1	27.6	0.03	0.176	superconducting
6	39	130.6	27.7	0.05	0.440	growing voltage

Chapter 3

Numerical Simulation

3.1 Differential equation

3.1.1 Conduction equation

The thermal behaviors of solids are governed by the transient conduction equation.

$$\rho c \frac{\partial T}{\partial t} = \nabla(k \nabla T) + G(x, y, T) \tag{3.1}$$

Because k is strongly temperature dependent, especially at cryogenic temperatures, the equation becomes non-linear, but derivation of Fourier's law of conduction shows that derivatives of k can be eliminated. Using the divergence theorem eliminates one of the gradients and creates a volume integral on the left that can be interpreted as the change in internal energy, E. Expanding the remaining gradient leads to Fourier's law of conduction and also shows that the cylindrical form, with an annular volume, is identical in form to the Cartesian version.

$$\begin{aligned}
\rho c \frac{\partial T}{\partial t} &= \nabla(k \nabla T) + G(x, y, T) && (3.2) \\
\int_V \rho c \frac{\partial T}{\partial t} &= \int_V \nabla(k \nabla T) + \int_V G && (3.3) \\
\frac{d}{dt} \int_V \rho c T &= \oint_S (k \nabla T) + \int_V G && (3.4)
\end{aligned}$$

$$\frac{dE}{dt} = \oint k\left[\frac{\partial T}{\partial r}\hat{r} + \frac{1}{r}\frac{\partial T}{\partial \phi}\hat{\phi} + \frac{\partial T}{\partial z}\hat{z}\right]\cdot d\vec{A} + \int_V G \tag{3.5}$$

$$\frac{dE}{dt} = -kA_{r_1}\frac{\partial T}{\partial r} - kA_{z_1}\frac{\partial T}{\partial z} - kA_{r_2}\frac{\partial T}{\partial r} - kA_{z_2}\frac{\partial T}{\partial z} + \int_V G \tag{3.6}$$

$$\frac{dE}{dt} = q_{boundary} + q_{internal} \tag{3.7}$$

The negative signs arise from the difference in convention for +/− at the boundaries between the surface integral and the gradients. [14, p. 23] Care must be taken to use the proper sign on energy entering or leaving a boundary based on the temperature gradient.

3.1.2 Boundary conditions

For the overall experimental setup, the boundaries are the copper attachment to the cold head and the physical limits of the magnet and coilform. The copper attachment to the cold head supports a fixed heat flux as a function of temperature, based on the performance of the cryocooler. For the duration of these pulse experiments it is assumed to be isothermal. The conductor nodes allow heat flow to the surrounding coil form, or there is a code switch to model the coil as adiabatic. Radiation heat input could be included at the outer edge of the coil, but is neglected in this modeling. Using Fourier's law of conduction in one dimension the flux based adiabatic boundary condition is $q = 0$, and $q = f(T)$ for the edges next to the coil form, creating Neuman boundary conditions which are homogeneous at the adiabatic interfaces and nonhomogeneous at the walls of the coilform where the cryocooler removes heat.

3.2 Material Properties

Part of the accuracy of any working simulation is its agreement with the physical parameters being represented. Most of the material properties included in the diffusion equation are temperature dependent. In addition there is also a dependency on magnetic field in the case of the silver conductor.

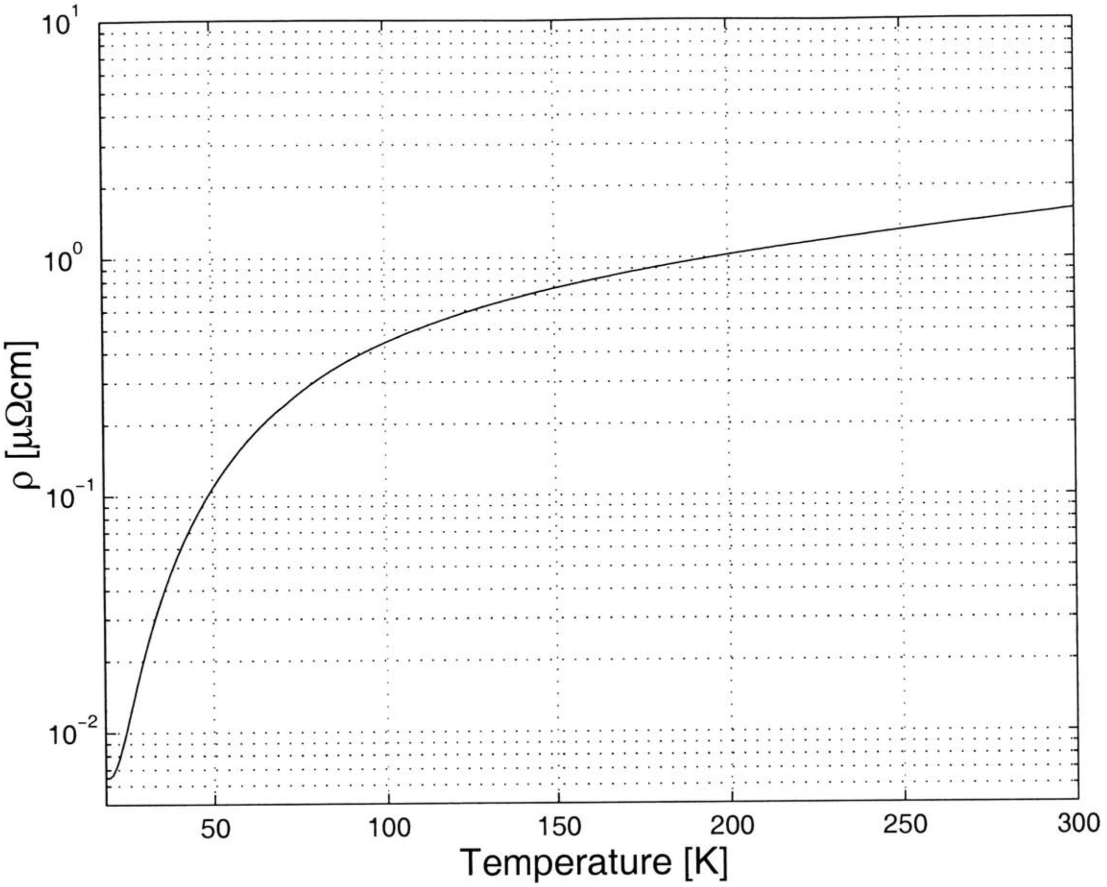

Figure 3-1: Electrical Resistivity of Silver

3.2.1 Silver

Electrical Resistivity

The resistivity of silver with zero background field, $\rho_{Ag,0}$, is a function of temperature as shown in figure 3-1. The data has been correlated by Hun [9] into a piecewise 3rd order polynomial:

Table 3.1: Correlation Coefficients for the resistivity of silver in 0 field.[9, p. 69]

Temperature range [K]	m_0 [$\mu\Omega$cm]	m_1 [$\mu\Omega$cm/K]	m_2 [$\mu\Omega$cm/K^2]	m_3 [$\mu\Omega$cm/K^2]
4-70	0.069136	-0.006714	0.00019844	-9.728×10^{-7}
70-300	-0.34145	0.0094905	-1.9905×10^{-5}	3.2803×10^{-8}
300-1000	1.6	0.005	0	0

Table 3.2: Correlation Coefficients of the Kohler function for silver.[9, p. 70]

T Range [K]	m_0	m_1
4-200	0.00014736	1.5838

$$\rho_{Ag,0} = m_0 + m_1 T + m_2 T^2 + m_3 T^3 \tag{3.8}$$

where the coefficients, m_0, m_1, m_2, m_3, are given in Table 3.1. The large fluctuations in the resistivity of silver in the range 20 K to 100 K show the importance of accurate modeling in this range, especially since this is also the range of current sharing for BSCCO-2223.

Magnetoresistive Effect

The resistivity of silver has a strong dependence on magnetic field, known as the magnetoresistive effect. According to Kohler's rule:

$$\frac{\rho_{Ag}(B,T) - \rho_{Ag,0}(T)}{\rho_{Ag,0}(T)} = f\left[\frac{B}{\rho_{Ag,0}(T)}\right] \tag{3.9}$$

where $\rho_{Ag}(B,T)$ is the resistivity of silver in a magnetic field B, and at temperature T. Figure 3-2 is a Kohler plot for pure silver(RRR=700), showing the relationship between a normalized change in resistivity due to magnetic field (the left hand side of Equation 3.9), the y axis, versus the product of field and normalized zero field resistivity, the x axis, given by:

$$x = B \times \frac{\rho_{Ag,0}(273K)}{\rho_{Ag,0}(T)} \tag{3.10}$$

The curve is piecewise approximated by the following equation:

$$y = m_0 x^{m_1} \tag{3.11}$$

where the coefficients, m_0 and m_1, are listed in Table 3.2 for x less than 200. We do not expect x to be greater than 200 because the test coil operates with zero background

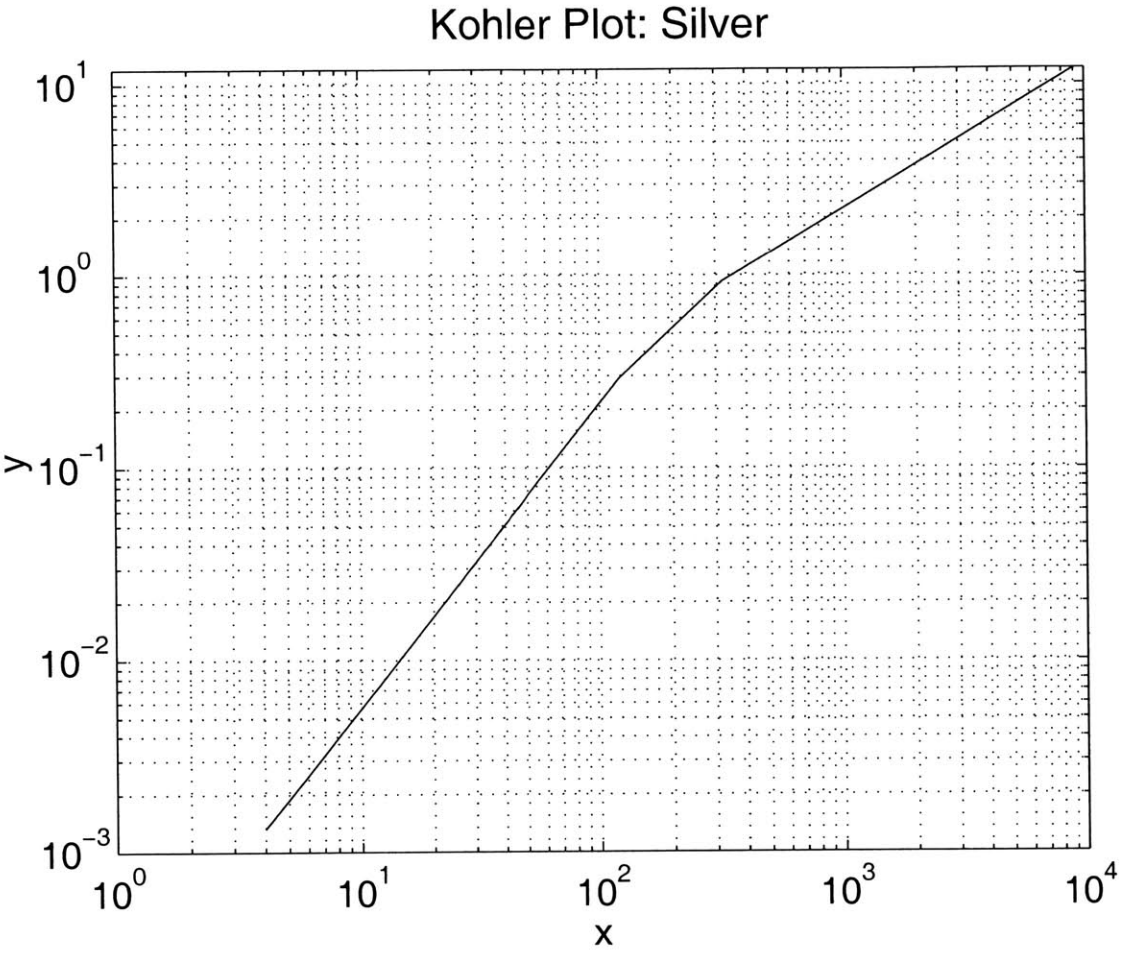

Figure 3-2: Kohler plot for silver(RRR=700)

field.

Thermal Conductivity

Figure 3-4 on page 34 shows the thermal conductivity of silver versus temperature in zero magnetic field, $k_{Ag,0}$. The data are from Bellis and Iwasa [16]. As with resistivity, conductivity is strongly temperature dependent and shows more than an order of magnitude of variation within the temperature range 10-100 K. An exponential correlation that approximates the data is given by:

$$k_{Ag,0}(T) = 4.0 + 3.2T^2 e^{(4.0-\frac{T}{4.5})} \tag{3.12}$$

for the temperature range 10 K to 300 K. In Equation 3.12, $k_{Ag,0}(T)$ is in the unit of [W/cmK] and T is in [K]. Above 77 K, $k_{Ag,0}$ is roughly constant at 4.0 W/cmK as shown in Figure 3-4.

Magnetic field also affects thermal conductivity similarly to its effect on resistivity. The zero field conductivity in a non-zero magnetic field may be estimated using the Wiedmann-Franz law, which applies to normal metals:

$$k_{Ag}(B,T)\rho_{Ag}(B,T) = \Lambda T \tag{3.13}$$

where Λ is the Lorentz number, $(2.45 \times 10^{-8} W/\Omega K^2)$. Replacing ΛT with resistivity and thermal conductivity in zero fields yields:

$$k_{Ag}(B,T)\rho_{Ag}(B,T) = k_{Ag,0}(T)\rho_{Ag,0}(T) \tag{3.14}$$

and

$$k_{Ag}(B,T) = k_{Ag,0}(T)\frac{\rho_{Ag,0}(T)}{\rho_{Ag}(B,T)}. \tag{3.15}$$

However, the Lorentz number increases with magnetic field at a rate of approximately

0.05 per tesla. Therefore,

$$k_{Ag}(B,T) = (1 + 0.05B)k_{Ag,0}(T)\frac{\rho_{Ag,0}(T)}{\rho_{Ag}(B,T)}. \tag{3.16}$$

This relation applies reasonably to most conductive metals.

Specific Heat

The specific heat of pure metals may be expressed as the sum of an electronic (Sommerfield) contribution c_{v_1}, and a phonon (Debye) contribution, c_{v_2}. The electronic contribution is a function of temperature given by:

$$c_{v_1} = 1944\left(\frac{T}{\Theta}\right)^3 \tag{3.17}$$

where $c_{v_1}$is in units of [J/mol K] and Θ is the Debye temperature, equal to 220 K for silver[2]. The phonon contribution is a constant equal to the Dulong-Petit limit. This limit is approached above the Debye temperature and is given by:

$$c_{v_2} = 3R = 24.95 \tag{3.18}$$

where c_{v_2} is given in [J/mol K] and R is the universal gas constant. According to Dresner[2], the two may be combined into a single equation, yielding:

$$c_v = (c_{v_1}^{-n} + c_{v_2}^{-n})^{-1/n} \tag{3.19}$$

where n=0.85, $c_{v_1} = 17.75 \times 10^{-6}T^3$ J/cm^{3}K, and $c_{v_2} = 2.426$ J/cm^{3}K for silver. Magnetic field has minimal effect on the specific heat of silver, c_{Ag}. The following function is used to calculate c_{Ag}:

$$c_{Ag} = [(17.75 \times 10^{-6}T^3)^{-0.85} + (2.426)^{-0.85}]^{-1/0.85} \tag{3.20}$$

which is the Dulong-Petit law [2] and agrees with experimental data. [15, 13] Heat

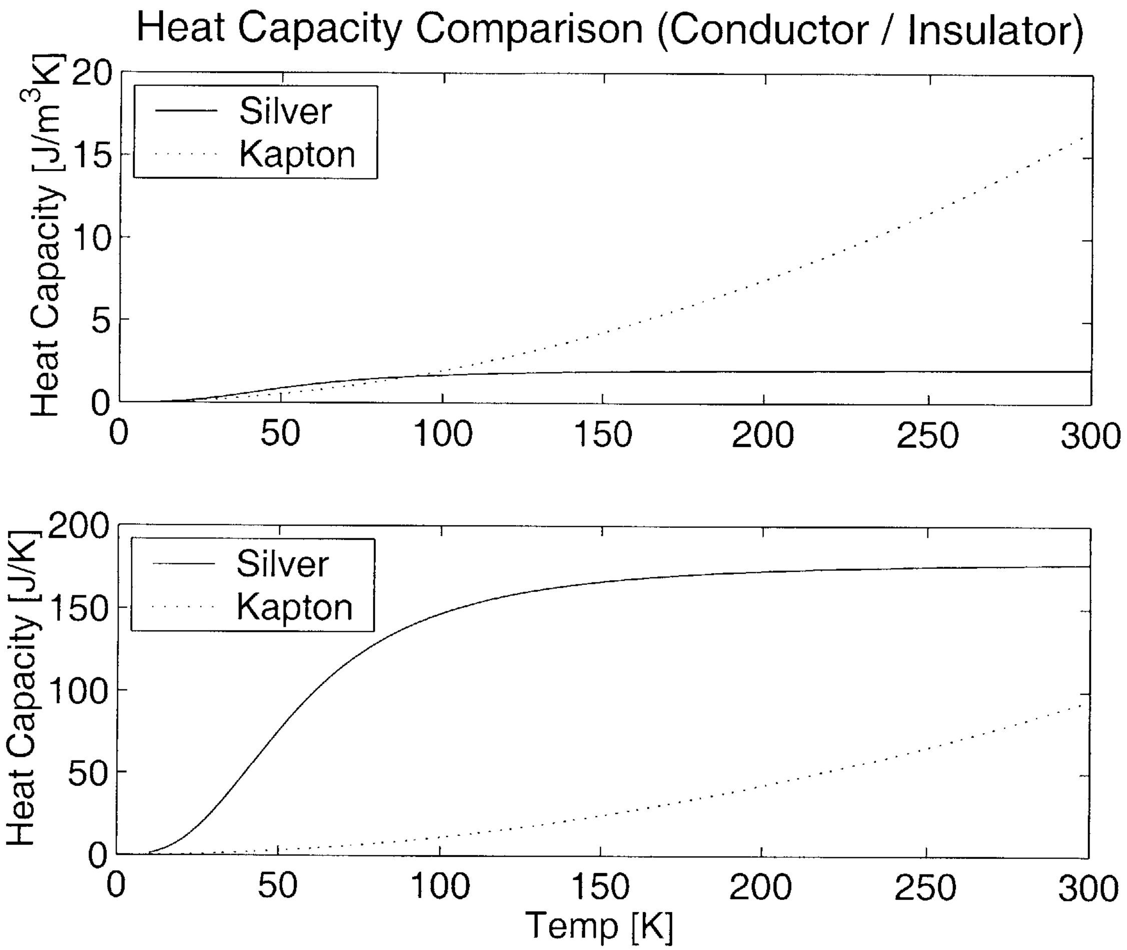

Figure 3-3: Heat capacity of conductor (silver) and insulator. The upper graph is the volumetric comparison while the lower is the absolute capacity based on the volume of each material with a single node.

capacity is found in Figure 3-3.

3.2.2 Copper

Although copper is used extensively in the coilform and support structures, there is not any within the winding or simulated coil. For the purpose of simulation the properties of copper are only relevant at the boundaries. Improving the accuracy of the material properties of copper would improve the simulation, but for simplicity only an approximation is used. Above 20 K the thermal conductivity of copper is similar to that of silver and therefore the same formula is used to calculate copper properties, without the inclusion of the magnetoresistive effects. [7, p. 385]

3.2.3 BSCCO-2223

Thermal Conductivity

There are few references listing the thermophysical properties of BSCCO-2223. The thermal conductivities of undoped, and silver doped BSCCO-2212, reported by Matsukawa et. al.[10], are on the order of 0.001 W/cmK. By comparison to Figure 3-4, we see that the thermal conductivity of silver is at least 2 orders of magnitude larger than the conductivity of BSCCO-2212. The thermal conductivity of the conductor, k_{cd}, is modeled as the volume fraction of silver, x_{Ag}, multiplied by the thermal conductivity of silver. This approximation is valid because the the silver and BSCCO are conducting heat in parallel and the silver component dominates the combination.

Specific Heat

Specific heat data for BSCCO-2223 is as sparse as the conductivity data. The properties given by Iwasa [7] for BSCCO-2223 and silver at 120 K show that the volumetric specific heat of BSCCO-2223 is about half c_{Ag}. For a silver fraction $x_{Ag} = 0.70$, we estimate the volumetric specific heat of the conductor, c_{cd}, as $0.85c_{Ag}$.

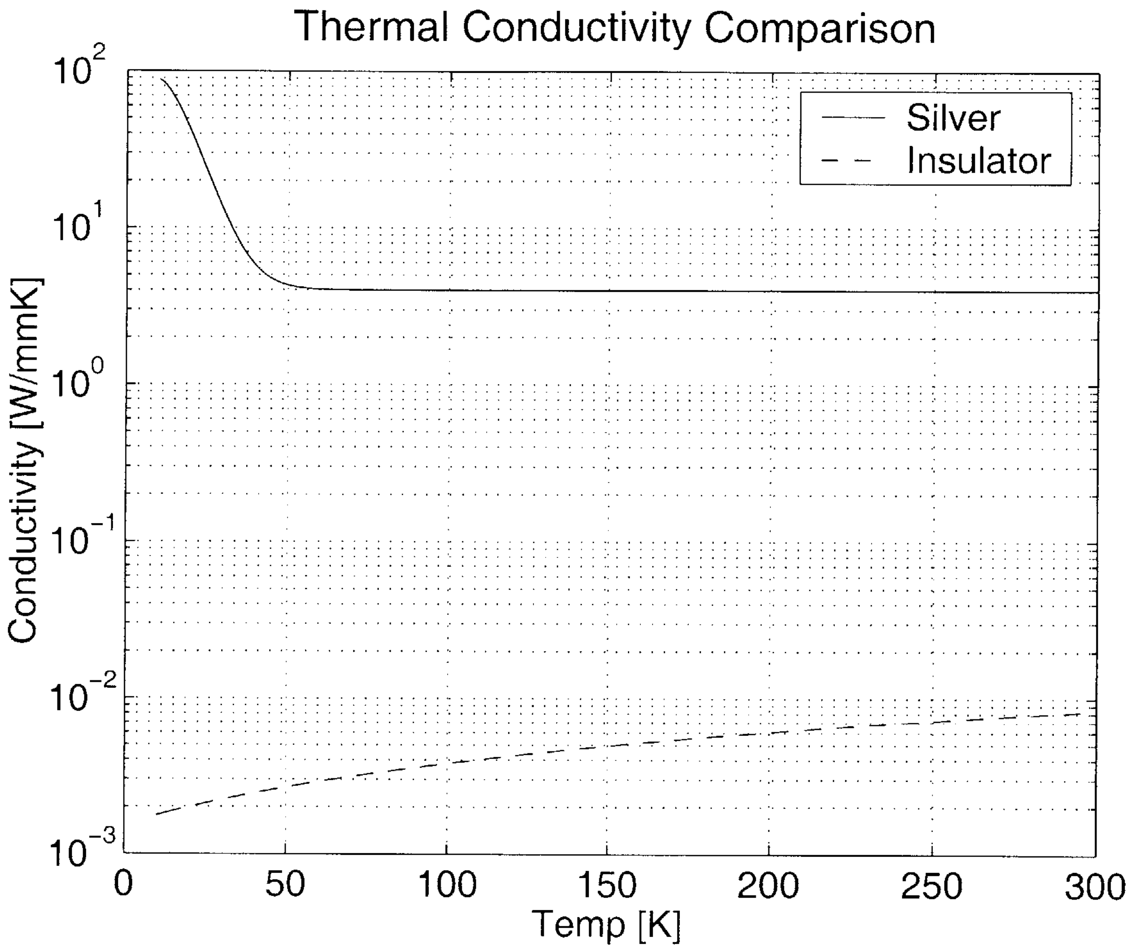

Figure 3-4: Thermal conductivities of silver and insulator

Critical Current

The critical current in the conductor, I_c, is not correlated as a function of temperature prior to winding the coil. Ideally, a slowly ramped current at a fixed temperature, repeated at various temperatures would provide accurate data regarding the critical currents. Attempting this presents several challenges. The coil is wound with 2 separate specifications of conductor in the inner and outer coil. Although voltage measurements across the coils can easily be taken separately, as the outer coil enters the normal regime it heats up and skews the temperature in the neighboring layers. The uniformity of the coil temperature can only be estimated, since it is impossible to place thermocouples into the interior of the magnet. All temperature assertions are made as an interpolation between the coilform temperature and the thermocouples attached to the outer layer of the coil. Despite the difficulties, such ramped current tests were performed to ensure that the coil was operating near the specifications given by the test coil manufacturer. Using the supplied specifications, and observed behavior, the critical current is calculated by quadratic approximation.

Electric field in a superconducting material is related to transport current by the following function:

$$E = E_c \left(\frac{I_t}{I_c}\right)^n \tag{3.21}$$

where E_c is the electric field at the critical current, I_c, typically defined as 1 μV/cm, and n is the index number. Sumitomo measured fields of 0.1 μV/cm and 1.0 μV/cm at 77 K as 19.9 A and 23.7 A. I_c vs. T data found in [7] indicates the relation is approximately linear with a slope of -1.82 A/K, and a T-intercept of 90 K. Using the Sumitomo data I_c as a function of T with a critical field of $E_c = 1$ μV/cm gives:

$$I_c = -1.82T + 163.8 \tag{3.22}$$

The critical temperature, T_c, is estimated by setting the transport current, Ic, to zero and solving for T, which gives $T_c = 89.3$ K. When cooling down the coil with a constant 1 A current the voltage across the coil dropped to unmeasurable levels at

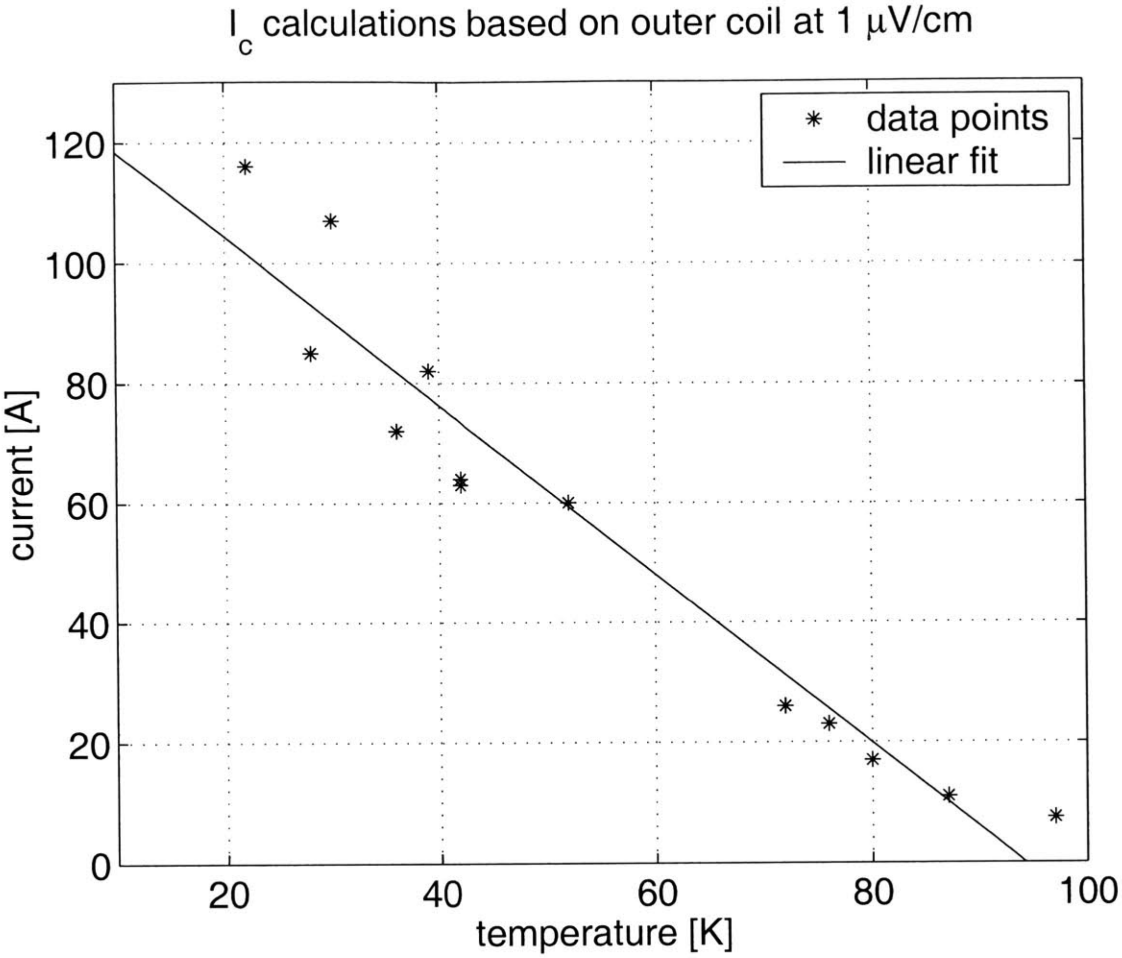

Figure 3-5: Critical current measurement data from test coil

just under 90 K.

When testing the coil, and in running experiments these formulas overestimated the heat generation, in other words they underestimated the critical current, at low temperatures. As a way of verifying the manufacturer specifications critical current tests were performed on the whole magnet. The current was ramped up at less than 0.5 V/s and the currents and voltages were recorded. Figure 3-5 shows the results of those rough measurements with the critical current being calculated based on an outer coil voltage of 1 μV/cm. This data also demonstrates the difficulty in obtaining consistent and accurate temperatures. In other tests, with pulses to the magnet, the critical current crossover point is above 160 A at 20 K. Such information as this

Table 3.3: Quadratic Fit Points for Critical Current Function

Temperature	Inner I_c	Outer I_c
20	161.0	120.0
39	140.0	118.0
89.5	0	0

Table 3.4: Thermophysical Properties of Kapton

Property	Temperature [K]	m_0 [W/cmK]	m_1 [W/cmK^2]	m_2 [W/cmK^3]
k_{kapton}	20-300	0.0015335	0.0000227	0
	[K]	[J/cm^{3}K]	[J/cm^{3}K^2]	[J/cm^{3}K^3]
c_{kapton}	20-300	-0.016000	0.002074	1.7738×10^{-4}

prompted Haid to use the approximation,

$$I_c = -2.27T + 203.5 \tag{3.23}$$

These matched his experiments, but showed discrepancies in this simulation. A quadratic fit was created with 3 observable data points and iterating the simulation in an attempt to match the simulation to the experiments. The final data points used are summsummarizearised in Table 3.3 and present an increased critical current capacity in the 40-60 K range relative to th linear approximation. Additional consideration is given to this issue in the discussion of results.

3.2.4 Kapton

Figures 3-3 and 3-4 show the heat capacity and thermal conductivity respectively of Kapton as used in the simulation. Thermal properties for Kapton are not directly available, but properties of organic materials do not vary significantly, so the data used are from epoxy. [9, 13, 15] Magnetic field does not affect the properties and polynomial correlations are similar to Equation 3.8 on page 28 using the data found in Table 3.4.

3.3 Power dissipation

Internal heat generation g(T,x) comes from transport current flowing down a voltage gradient:

$$g(T,x) = I_t \frac{\partial V}{\partial x} \tag{3.24}$$

When the critical current is larger than the operating current there is no voltage gradient and therefore no heat generation. Once the operating current reaches the critical level current sharing begins, where the superconductor carries I_c and the remainder of the operating current is carried by the metal matrix, in the case of BSCCO-2223, silver. The voltage drop across the conductor is related to the current in the matrix by:

$$\frac{\partial V}{\partial x} = I_m \frac{\rho_m}{x_m A_{cd}} \tag{3.25}$$

where A_{cd} is the area of the conductor, and $x_m A_{cd}$ is the cross sectional area of the matrix metal, that metal having resistivity ρ_m. The matrix current, I_m, is therefore $I_t - I_c$. Combining Equation 3.24 with Equation 3.25 leads to:

$$g(T,x) = I_t(I_t - I_c)\frac{\rho_m}{x_m A_{cd}} \tag{3.26}$$

In the normal state the critical current is reduced to zero. Assuming that the resistivity of the normal state superconductor is much larger than the matrix metal, when $I_t > I_c$ then:

$$g(T,x) = I_t^2 \frac{\rho_m}{x_m A_{cd}} \tag{3.27}$$

The overall modeling of the internal heat dissipation within the test coil may be expressed as the following piecewise function:

$$g(T,\, I_c) = \begin{cases} I_t(I_t - I_c)\frac{\rho_m}{x_m A_{cd}} & \text{if} \quad T < T_c \text{ and } I_t > I_c \\ I_t^2 \frac{\rho_m}{x_m A_{cd}} & \text{if} \quad T \geq T_c \\ 0 & \text{if} \quad T < T_c \text{ and } I_t \leq I_c \end{cases} \tag{3.28}$$

Within the current sharing region, Equation 3.23 is substituted for I_c in Equation 3.28.

3.4 Numerical model

3.4.1 Justification of a 2D Cartesian system

The magnet design can approximately be considered axially symmetric. The only correction to this assumption is the net vertical flow of heat through the helical path of the conductor. The differential equation is modeled in two dimensions with the possibility of correcting terms to increase the accuracy resulting from this helical conduction along the length of the conductor.

The heat equation Cartesian coordinates with constant physical properties is

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = \frac{1}{a_t}\frac{\partial T}{\partial t} \tag{3.29}$$

and the axially symmetric form of the heat equation in cylindrical coordinates is

$$\frac{\partial^2 T}{\partial r^2} + \frac{1}{r}\frac{\partial T}{\partial r} + \frac{\partial^2 T}{\partial z^2} = \frac{1}{a_t}\frac{\partial T}{\partial t} \tag{3.30}$$

If the x and y directions are equated with the r and z directions it can be seen that the only difference is the term $\frac{1}{r}\frac{\partial T}{\partial r}$. By comparing the relative magnitudes of these terms the appropriateness of the two dimensional Cartesian scheme can be judged. Table 3.5 explains the meaning of each term and gives a magnitude estimate relevant to the experimental setup. The ratio listed in Table 3.6 is each term relative to the radial term from the cylindrical system.

These magnitudes show the significance of terms in the second order heat equation. In the simulation the second order heat equation is not used because of the modification of the heat equation using the divergence theorem. Comparing Equation 3.1 and Equation 3.5 on page 25 show that the order has been decreased by approximating

Table 3.5: Heat Equation Terms Explanation

term	meaning	value
Δr	horizontal distance between conductors	0.0575 cm
R	average coil radius	52.9 cm
Δz	vertical distance between conductors	0.363 cm
τ	time step	0.0001 s
$\frac{\rho c}{k}$	thermal properties	$0.04935 \frac{\text{s}}{\text{cm}^2}$

Table 3.6: Heat Equation Terms Magnitude Comparison

term	magnitude	ratio	value
$\frac{1}{r}\frac{\partial T}{\partial r}$	$\frac{\Delta T}{R\Delta r}$	1	1
$\frac{\partial^2 T}{\partial r^2}$	$\frac{\Delta T}{\Delta r^2}$	$\frac{R}{\Delta r}$	920
$\frac{\partial^2 T}{\partial z^2}$	$\frac{\Delta T}{\Delta z^2}$	$\frac{R\Delta r}{\Delta z^2}$	8.4
$\frac{1}{a_t}\frac{\partial T}{\partial t}$	$\frac{\rho c}{k}\frac{\Delta T}{\tau}$	$\frac{\rho c R\Delta r}{k\tau}$	1500

the second order differential heat equation over a finite volume to obtain a first order derivative integrated over a surface. This decrease in order simplifies the equations and makes the non-linearity easier to deal with, but introduces other complications.

3.4.2 Finite difference method

The diffusion equation

$$\frac{\partial u}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial u}{\partial x}\right) \tag{3.31}$$

is the most commonly used example of a parabolic equation. [11] When the diffusion coefficient, D, is constant a whole host of methods are easily applied to create numerical solutions. By using the divergence theorem to lower the order of our equation the basic second order equation has been reduced to a first order and different numerical complications must be dealt with. The second order equation,

$$\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2} \tag{3.32}$$

which is stable when differenced with a FTCS scheme (Forward Time, Centered Space),

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = D\left[\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}\right] \tag{3.33}$$

is now transformed into a much more finicky first order advection equation

$$\frac{\partial u}{\partial t} = -\frac{\partial \mathbf{F}(\boldsymbol{u})}{\partial x} \tag{3.34}$$

performed on the flux rather than the temperature. [3, p 173] That flux is computed using a first order approximation $\mathcal{O}(\Delta x)$, rather than the second order approximation $\mathcal{O}(\Delta x^2)$ achievable with the second order equation. The non-linear nature of the equation, the non-homogeneous heat generation term, and the multiple and discontinuous material types make accurately dealing with the real equation a project all by itself and necessitates the flux method. The FTCS scheme, which on this reduced order equation is

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = -v \left[\frac{u_{j+1}^n - u_{j-1}^n}{2\Delta x}\right] \tag{3.35}$$

where in this simplification $\mathbf{F(u)}$ in Equation 3.34 is simply vu. When replaced with a modification by Lax,

$$u_j^n \Longrightarrow \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) \tag{3.36}$$

the stable Lax method is obtained.

$$u_j^{n+1} = \frac{1}{2}(u_{j+1}^n + u_{j-1}^n) + \Delta t\, D \left[\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{(\Delta x)^2}\right] \tag{3.37}$$

These equations, which are all listed in one-dimensional form, are easily expandable to multiple dimensions and can incorporate the nonlinear temperature dependence of the material properties. [11, p. 834-855]

The diffusion problem also has boundaries in both time and space. Such problems are frequently termed initial value problems. Boundary conditions can be either Neumann, with the derivative at the boundary specified, or Dirchlet, with a set boundary value. [11, 14] In our case we will assume that the boundary conditions are not functions of time, although they could be modeled that way to account for the physical behavior of the system. The flux leaving the coil and entering the coilform is a function of the local temperature and the coilform temperature, which is assumed constant.

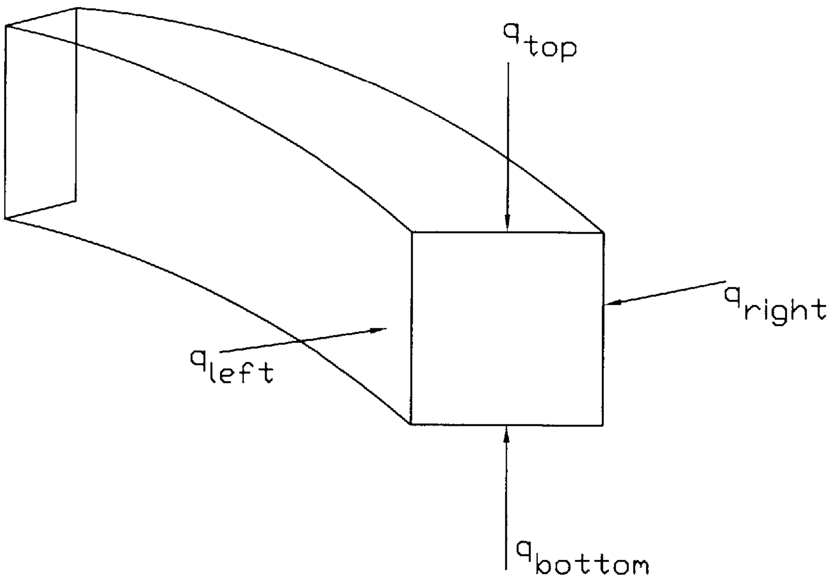

Figure 3-6: Flux entering the annular cell

Finite difference methods discretize the spatial domain into points and lump properties and variables at such points. These points can be node centered or cell centered depending on the boundary condition treatment. Because of the use of the flux conserved form of the diffusion equation, a cell centered orientation works best.

Differencing Equation 3.5 is a multi step process. Fourier's law of conduction in each direction can be obtained using numerical integration of the surface integral. Our cell-centered volume is annular with a rectangular cross section, as in Figure 3-6. Writing Equation 3.5 using the resulting flux yields

$$\frac{dE}{dt} = q_{r_+} + q_{r_-} + q_{z_+} + q_{z_-} + q_{internal} \tag{3.38}$$

where these q's are positive for flux entering the cell and the errors of the differencing method are related to the approximations of these quantities.

The internal energy generation is the easiest to calculate. Joule heating in the conductor due to the voltage across the silver matrix and the current through the conductor generates power according to Equation 3.28 on page 38. Since the cells are aligned with the conductors this energy is simply the volumetric power produced in

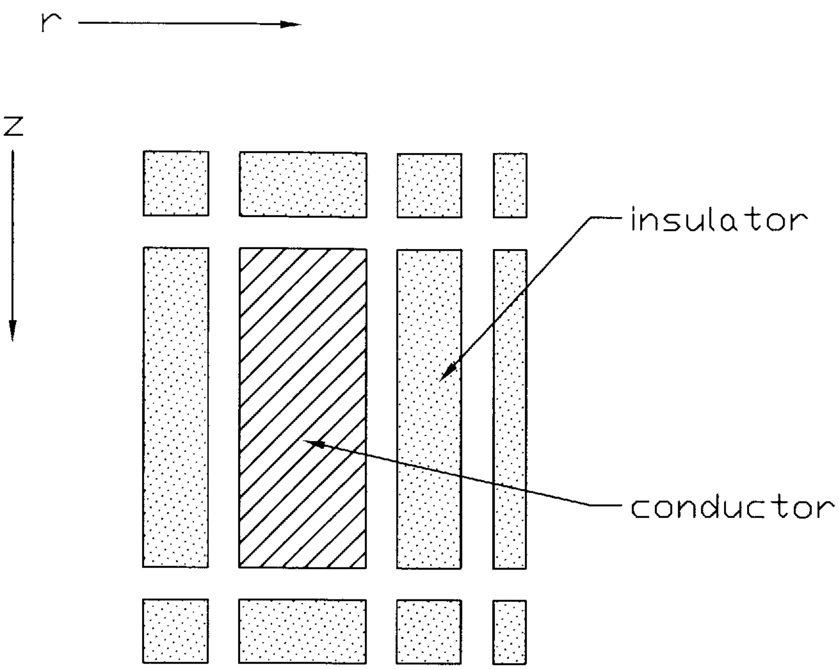

Figure 3-7: Single node sub-components. The r and z indices begin from 0 in the upper left corner, which is closest to the copper extension. This figure has a single (r, z) designation and the equivalent resistance is computed based on this model. The interface between rectangles is isothermal and there is no heat flow into or from the small squares at the corners, heat only flows along the main horizontal and vertical lines.

the cell.

Each q is calculated according to the standard thermal resistance method where the thermal resistance is defined from

$$q = \frac{\Delta T}{R} \qquad (3.39)$$

Using the analogy to electrical circuits, equivalent resistances can be pieced together with either adiabatic or isothermal assumptions along the network.[6] Figure 3-7 depicts a single node and the components used to calculate the equivalent resistance. Conduction is only along the main lines, and the conduction through the small squares at the extremities is ignored.

The final step in the differencing is to take the overall flux changes within a node and convert them into a new temperature,

$$T_{ij}^{n+1} = T_{ave}^{n} + \frac{\Sigma q \, \Delta t}{C_{ij}} \tag{3.40}$$

where C_{ij} is the heat capacity of the ij node.

3.4.3 Mesh Geometry

Each node is centered around a turn of conductor. This does not mean that the surrounding insulation has been ignored. The original model, the lower part of Figure 3-8, separated the cross section into nodes of a single material. Each conductor is surrounded by a layer of Kapton insulation on all sides and there is an additional layer of Kapton insulation between layers. With three layers of insulation between each conductor one turn of the magnet was represented by 12 nodes. With 240 turns total for the inner and outer coils the computation was prohibitive. In addition debugging such a complicated code and searching for instabilities was a challenge.

Trading in un-necessary complication for simplicity was helpful. Each conductor was converted to a single node and the effects of the intervening insulation between adjacent conductors were included in the resistance modeling.

3.4.4 Node to Node Conduction

The heat flux between nodes is calculated in two steps. The resistance between nodes is first calculated using the electrical analog with parallel and series resistance where each resistor is

$$R = \frac{l}{kA}$$

where l is the length of the resistive section, k is the conductivity of the section, and A is the area of conduction. Figure 3-8 shows the complication that can be applied to the resistance network. Based on the dimensions of the conductor and insulation, minimum dimension 0.23 mm vs. 0.025 mm, and the differences in conductivities discussed in 3.2.3 the flux conducted along the insulators and parallel to the central

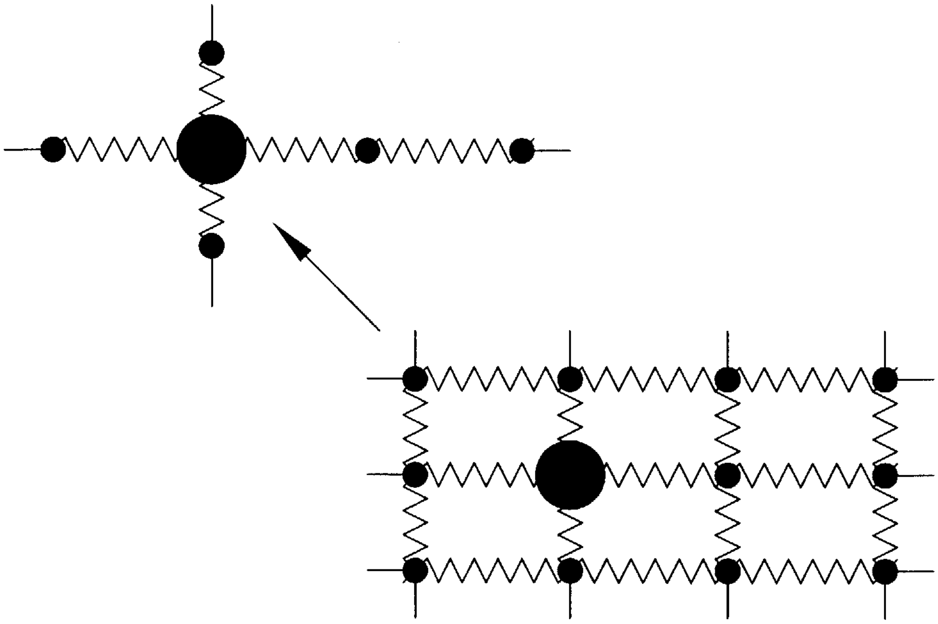

Figure 3-8: Resistance network with central conductor and surrounding insulation. The heat transfer along the insulation to the side of the conductor is negligible relative to the path in line with the conductor. Heat capacity of the network is lumped together and the whole node is considered isothermal.

path was assumed to be negligible. To confirm such an assumption the overall resistance network was entered symbolically in Maple V, a symbolic math manipulation program, and calculated. By expanding the resulting flux according to the insulation dimensions, the relative error associated with the conduction along the insulation could be deduced, which was less than 1%. The algebraically simplified expressions for the inverse of the resistance were then input into the simulation code.

3.4.5 Boundary Treatment

Various stages of evolution have passed in the implementation of this algorithm. From the initial complexity it has been simplified in an effort to stabilize the fluctuations at the boundaries. Adiabatic boundaries were used to debug the code and adjust the initial parameters. Once the adiabatic condition was removed the treatment of flux flowing to the coilform was similar to the treatment of the node to node conduction, but the isothermal boundary temperature was used.

3.4.6 Time Step

Explicit differencing schemes often have a stability condition that limits the choices of differencing parameters. Since the spatial dimensions are frequently fixed, time is the parameter adjusted as needed to achieve stability. For the simple two-dimensional Lax scheme the stability condition is

$$\Delta t \leq \frac{\Delta x}{\sqrt{2}|v|}$$

with v the maximum propagation velocity. [11] Here Δx is the minimum distance between nodes and the maximum propagation velocity can be found from

$$v_{max} = \left(\frac{\rho c_p}{k}\right)_{max} = \alpha_t$$

Using this condition and the simulation parameters leads to $\Delta t \leq 0.002$ s. An alternate condition is listed by Holman [6, p. 179] for the FTCS scheme as

$$\Delta t \leq \left[\frac{C_i}{\Sigma_j (1/R_{ij})}\right]_{min}$$

This condition is also used by Hun [9] and Haid [4] and it estimates the time step required as $\Delta t \leq 0.0005$ s. In this simulation the time step is set to 0.0001.

3.4.7 Scheme Implementation

The algorithm for the simulation is based on the Lax scheme discussed in Section 3.4.2. The temperature matrix is set up with an initially uniform temperature and the material properties are calculated at 0.1 K intervals and stored in a table for faster execution. All adjustable parameters are included as constants definitions at the beginning of the code to simplify variations. Separate functions make the code more readable and flexible. Output of voltage and temperature data are to ASCII files formated for easy input into Matlab, where post-processing and graphical manipulation are carried out.

Chapter 4

Results Discussion

4.1 Experimental Observations

In the Introduction anticipated cases were listed. They were:

- A runaway quench. This will damage the conductor if the current load is not removed.
- Non-quench. The reduced current following the pulse may be within the conductors superconducting range at the operating condition, allowing diffusive cooling to the cryogenic heat sink.
- A thermal recovery. Joule dissipation removal lowers the conductor temperature.
- A pseudo stability. Heat removal approximately matches the heat generation and a pseudo-steady state is observed. In reality this state eventually leads to a quench or recovery.

All of the above phenomena were observed, some more easily than others, and the 6 trials outlined in this paper are representative of those experiments. Through all of the efforts many insights were gained and some assumptions had to be modified.

Table 4.1: Trial Data Summary

Trial	Temperature [K]	Pulse Length [s]	Peak Current [A]	Stepped Current [A]
1	20	3.84	123.9	36.0
2	20	2.82	135.0	51.9
3	19	3.28	159.2	29.4
4	39	3.06	120.2	27.6
5	39	2.94	128.2	27.6
6	39	3.00	130.6	27.7

4.1.1 Observed Cases

Pulsing the magnet repeatedly while raising the current with each successive trial showed the cross over point at which resistive voltage was measurable. The difference in superconducting properties of the inner and outer sections of the magnet made it nearly impossible to induce any lasting quench in the inner coil without destroying the outer one. This may be an intentional design to prevent an internal burnout, but it limits our ability to isolate phenomena and it complicates both prediction and simulation. The details of the 6 trials are summarized in Table 4.1. Graphical plots of the voltage traces are in Figures 4-1, 4-2, 4-3, 4-4, 4-5, 4-6 on pages 49 to 54. The spikes at the beginning and end of each pulse are due to inductance and are discussed in Section 4.1.2. Trial 3, shown in Figure 4-3, shows the experimental observation that the critical current of the inner coil was above 160 A. Currents larger than this produced voltages across the inner coil and caused heating.

Interesting differences exist at the two temperatures represented. At 20 K it was difficult to get a run away quench. The inner coil never maintained a voltage beyond the initial pulse and the outer coil, even when allowed to heat up significantly, was manageable with manual shut off procedures. Such high currents do increase the risk of damage, for once a quench begins it progresses rapidly at these currents. Chyu and Oberly report that with moderate quench initiation energies on the order of 10 to 100 mJ silver sheathed tape is remarkably stable "especially for low-temperature operation; however, a strong energy pulse can result in a nearly spontaneous local melting inside the superconductor." [1] They also report on the quench protection

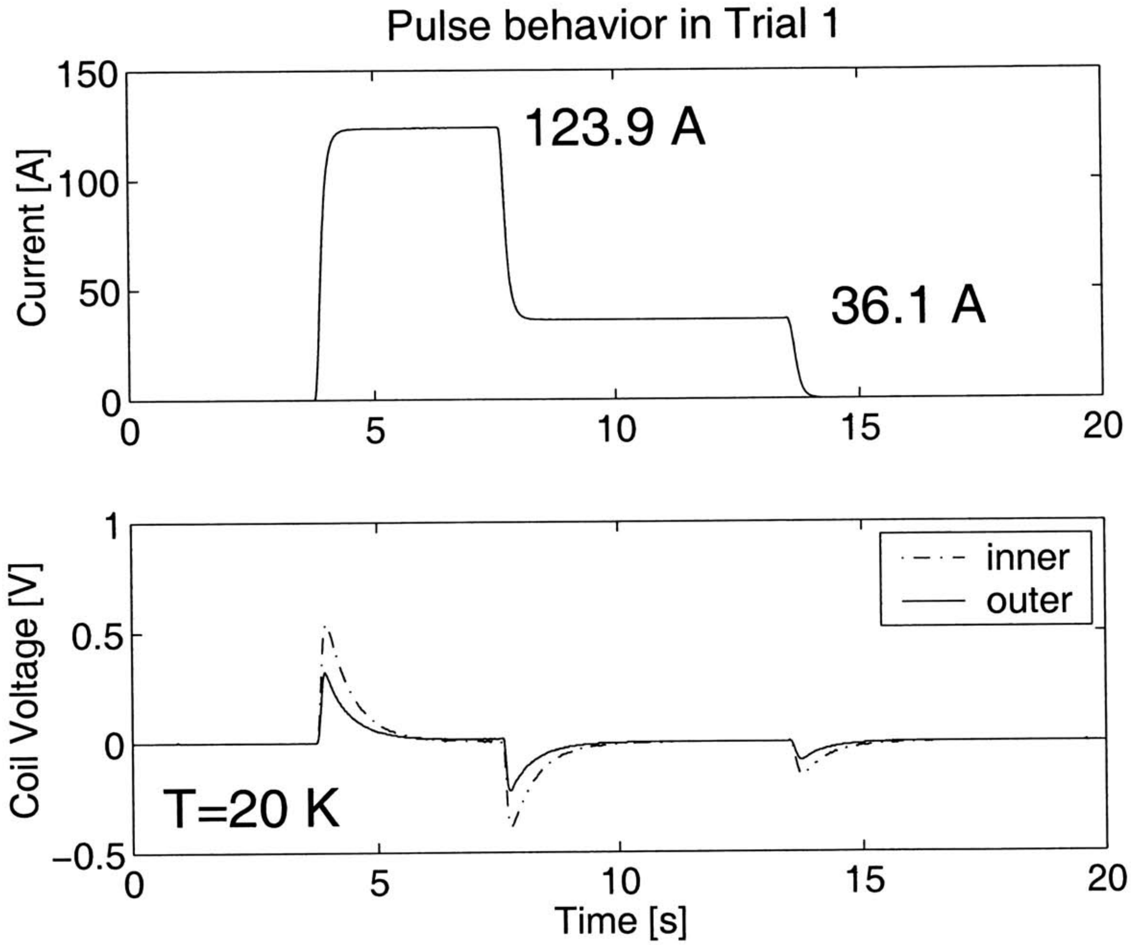

Figure 4-1: Trial 1 current and voltage data.

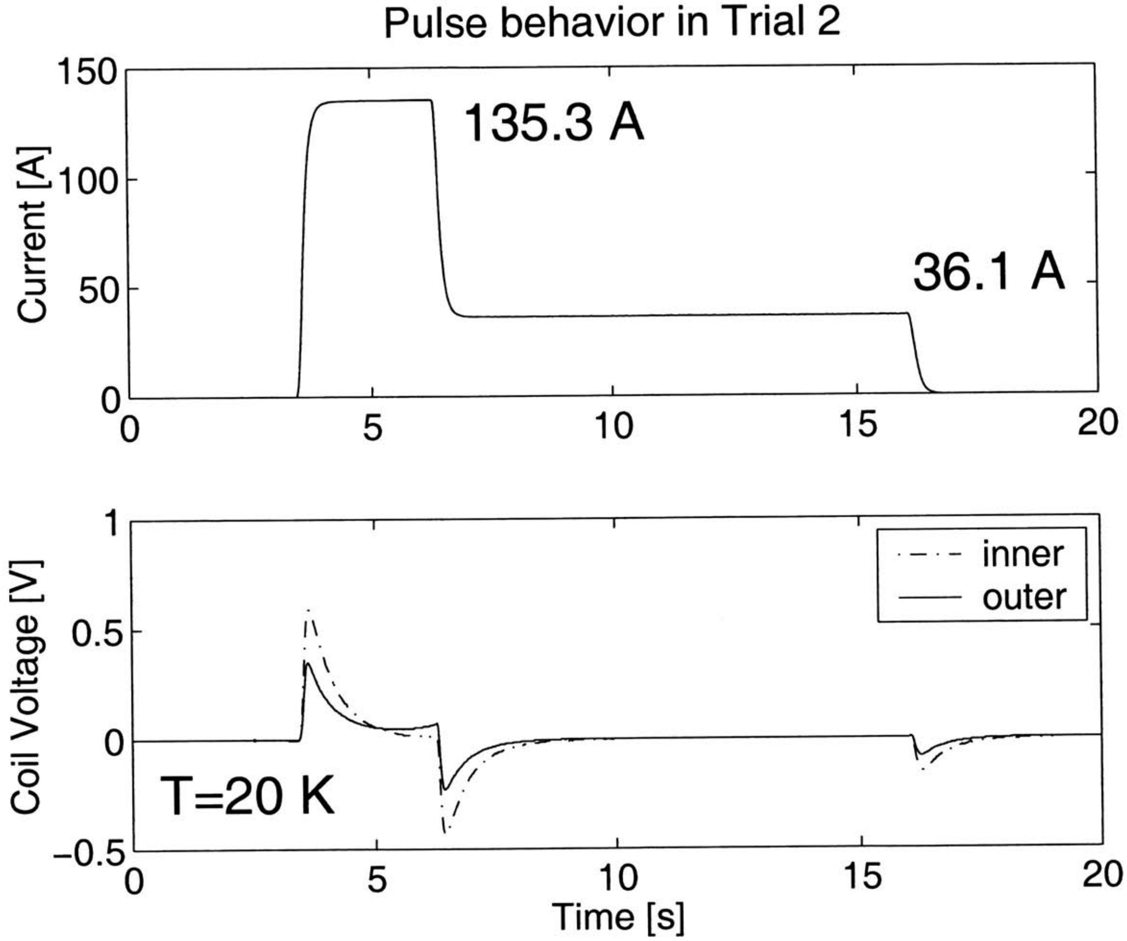

Figure 4-2: Trial 2 current and voltage data.

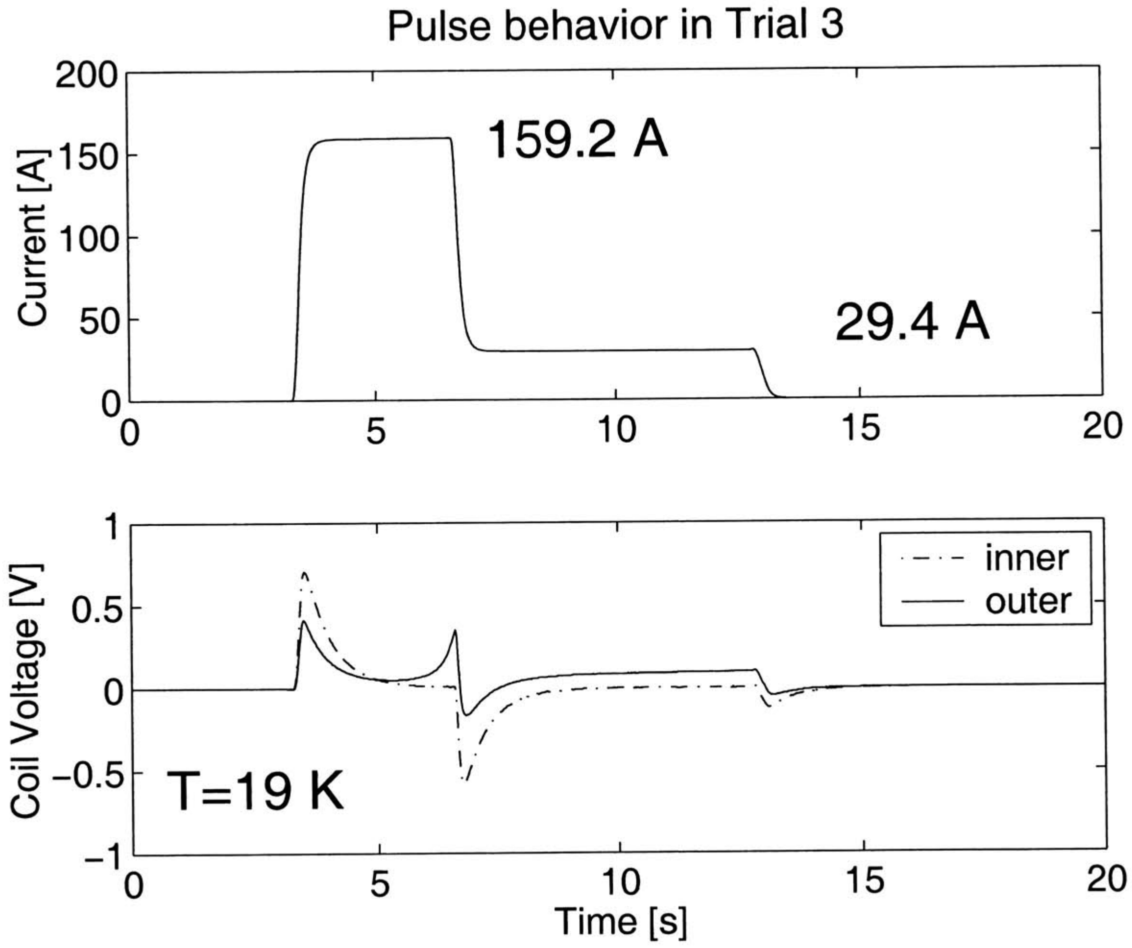

Figure 4-3: Trial 3 current and voltage data.

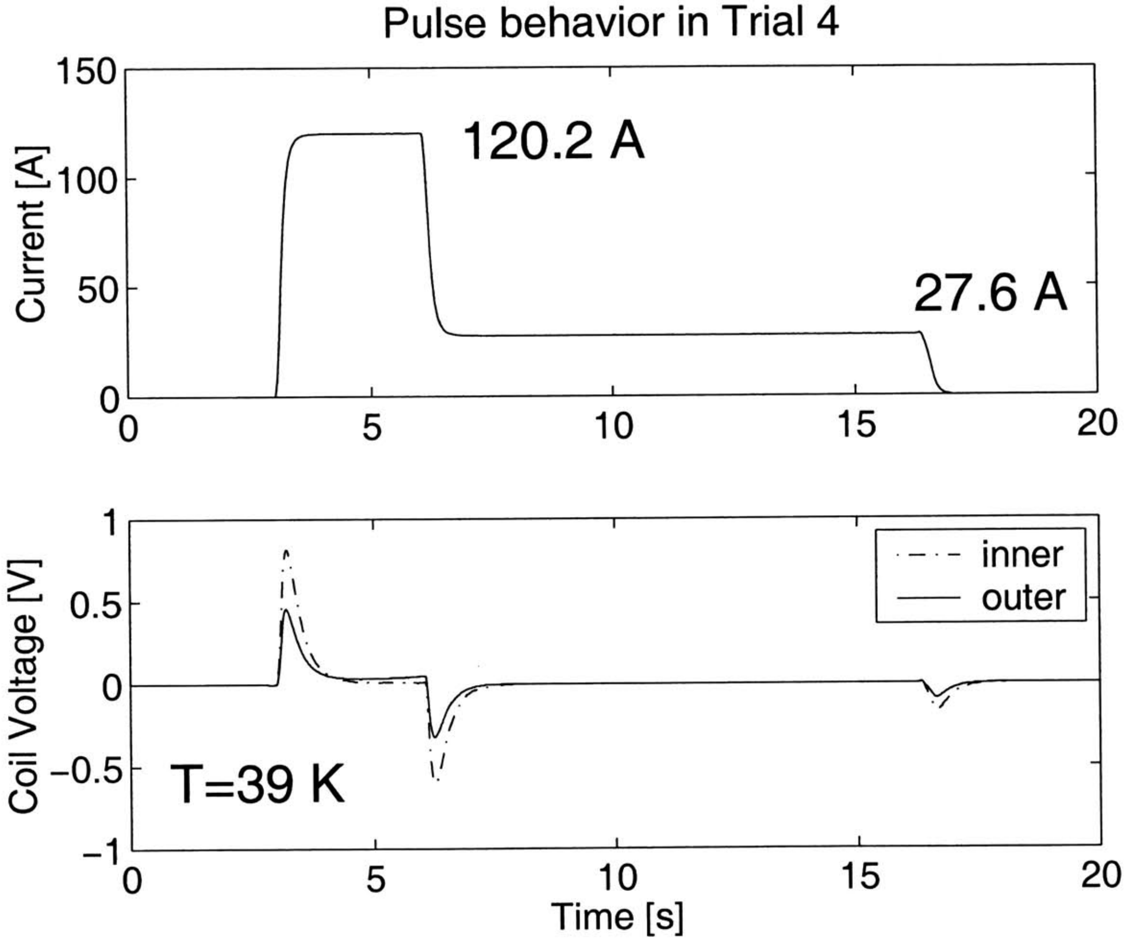

Figure 4-4: Trial 4 current and voltage data.

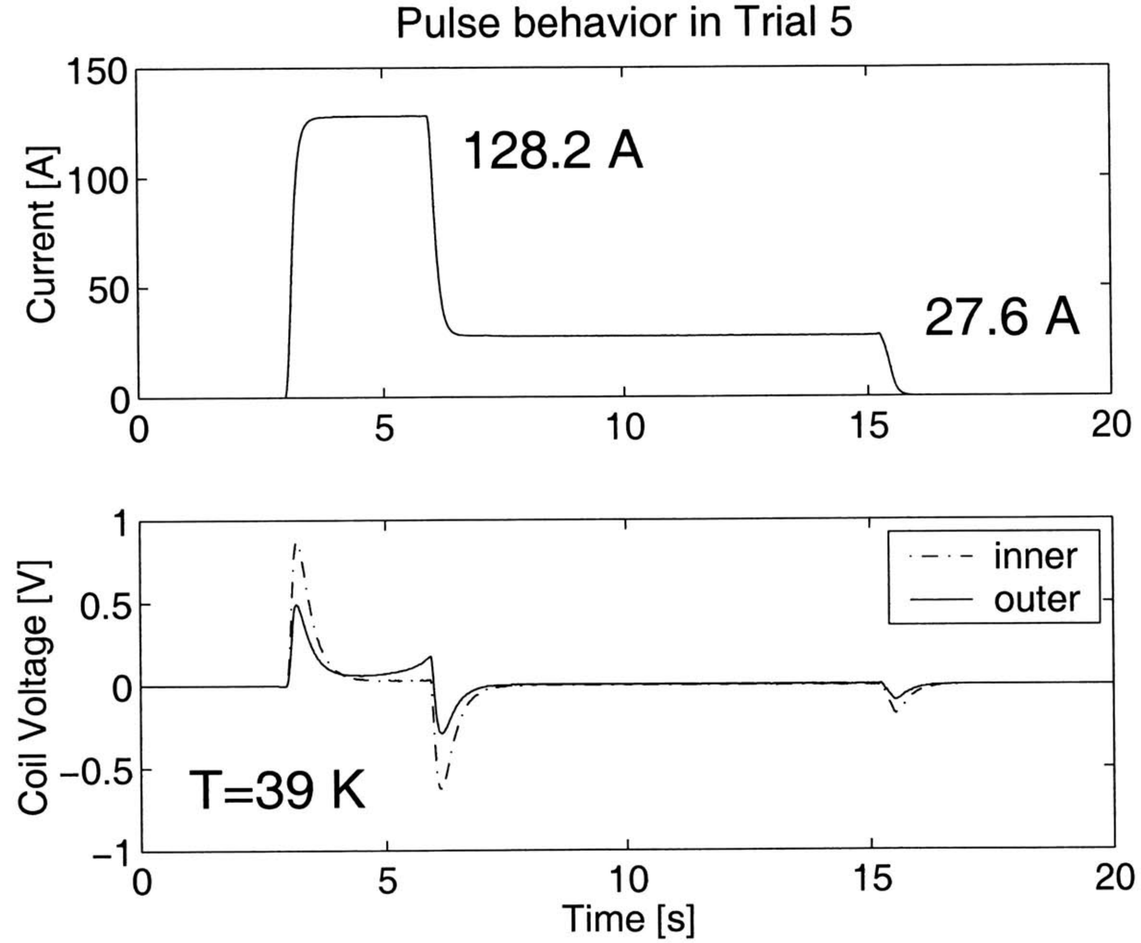

Figure 4-5: Trial 5 current and voltage data.

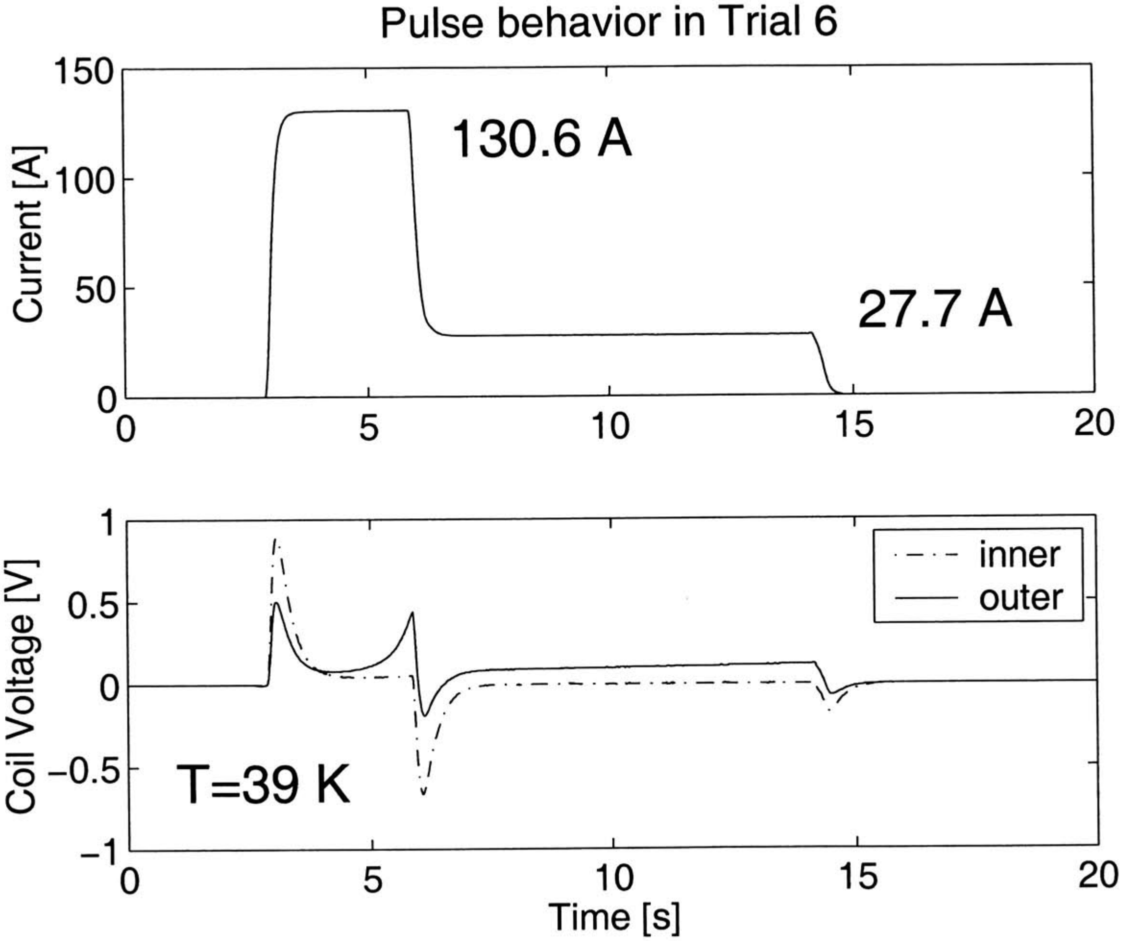

Figure 4-6: Trial 6 current and voltage data.

properties of HTS in low temperature operation. At 20 K operation they show that temperature graphs can be misleading due to nuances in material properties that foster recovery despite heat generation. As the operating temperatures increase the tendency to recover drastically decreases. The heat capacity of silver, the metal sheath around the BSCCO conductor, increases drastically with temperature. In Trials 5 and 6 the observable voltage on the inner coil betrays the smaller operating range available at these initial temperatures.

4.1.2 Inductive Voltage

Electrical circuits have inductance. In the case of a solenoid this inductance is very pronounced and readily observable when currents within the solenoid change. An Inductive voltage V is related to the inductance, L,by

$$V = L\frac{dI}{dt} \tag{4.1}$$

where I is the current. The voltage spikes seen in Figures 4-1 to 4-6 are due to inductance. Figure 4-7 shows a comparison of the data collected from the inner coil during the pulse of trial 3 compared to the current data and the calculated dI/dt. The data was collected at 50 Hz and dI/dt was calculated by differencing the offset data and dividing by Δt. Unfortunately, non-linearities made it impossible to subtract the affect of these voltages from the trial data.

4.1.3 Problems

The problems encountered with experimentation fall into 3 categories

- temperature measurements
- electrical shorts
- burn-outs

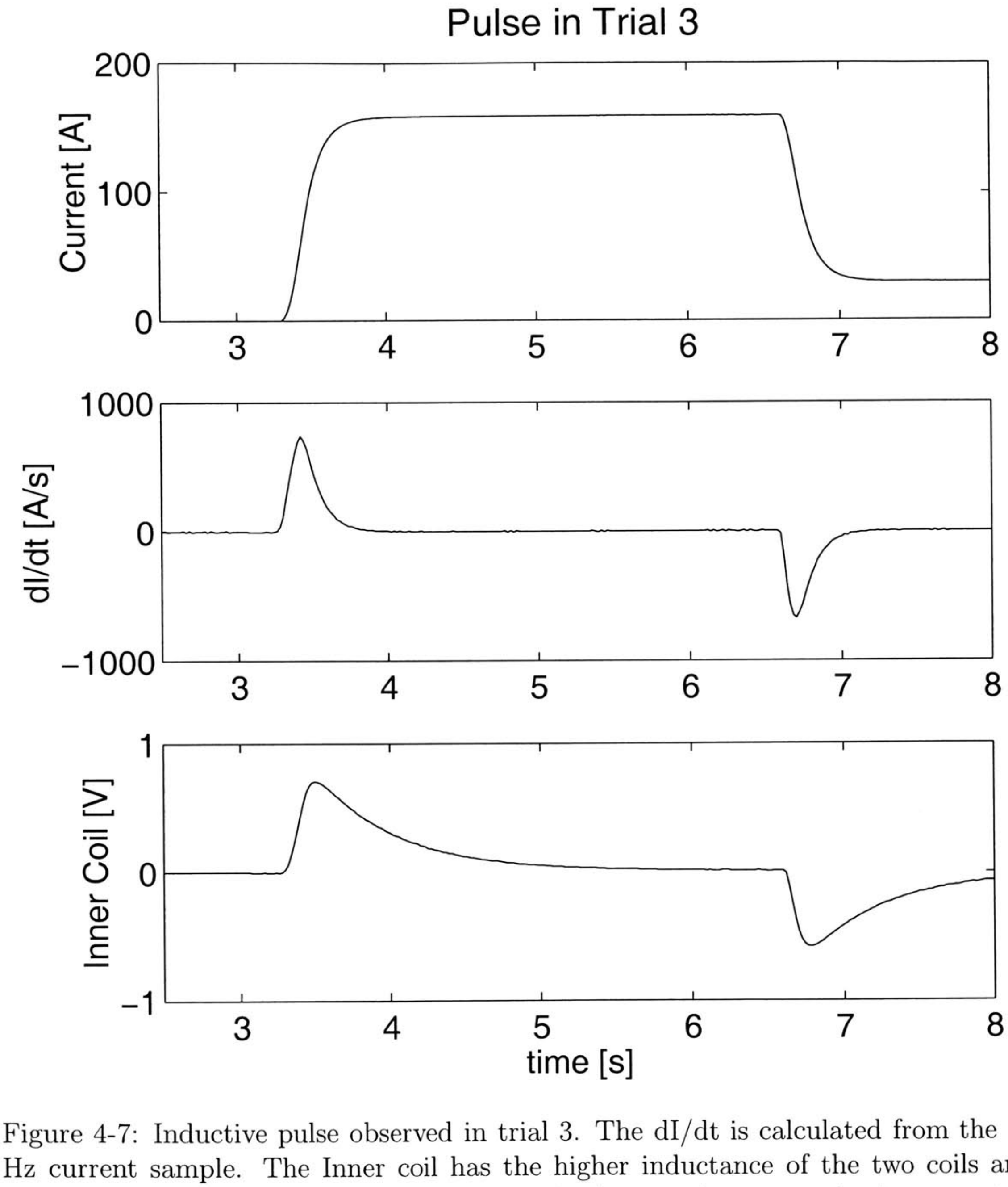

Figure 4-7: Inductive pulse observed in trial 3. The dI/dt is calculated from the 50 Hz current sample. The Inner coil has the higher inductance of the two coils and shows the induced voltage as the power supply changes the current in the magnet.

Of these 3, only the thermal measurement difficulties were not resolved satisfactorily. Patience and methodical work eliminated the shorts and burn-outs, although they both caused their share of problems.

The setup for the thermocouples involved slipping the thermocouple wire tip under the surface of Kapton insulation with thermal grease and then taping the wires against the magnet so they would not be pulled loose. In order to minimize inductive noise the wires for the leads were well twisted to decrease flux. All of the wires coming down the copper extension were wrapped around it and taped down to thermally anchor the wires and prevent heat input into the magnet. To reference the temperature for the thermocouples the opposite end from the one inserted under the Kapton was taped and greased to the base of the magnet. The thermo-diode attached to the coil form there would provide a reference temperature to measure the thermocouples from. Redundancy was included so that temperatures could be confirmed with multiple measurements.

Although the base temperature provided by the diode worked well throughout the experimental trials, it was affected by inductive pulses and had slow decay times in returning to a steady value. Determining which changes in measurement were due to thermal events and which were noise decreased confidence in the measurements. Additionally this unsteady reference temperature made clean traces from the thermocouple wires impossible. In spite of the non-presentability of the data, it was sufficient to check the coil temperature and estimate the temperature gradient between the coil form and the outer layer of the magnet.

Experience and lots of trial and error led to good estimates of the temperatures and the ability to determine from the data if the measurements were reliable or not. Those trials reported in this paper were those in which temperature measurements from several sources agreed. Even with good temperature measurements, the variation in temperature across the coil was unknown. With the super insulation in place and the cold space filled with liquid N_2 there is still the possibility that parts of the magnet were not at uniform temperature.

To redo the experiment effectively a way of either measuring or calculating the temperature of the magnet is needed. A more permanent and stable thermal anchor will be required for the thermocouples as well as a way of eliminating the inductive noise from the diode.

With so many parts of the experimental setup made of copper it was important to electrically insulate all of the wires on the current path. After one of the burn-outs the magnet had been prepared and the process of re-measuring the room temperature resistance and checking the connections was underway. Some unexpected values showed electrical shorts between the coil form, one of the voltage taps, and conductor. Drawing diagrams and calculating expected resistances from different points helped to isolate the sort. One short was at the top of the copper extension where the edge of the HTS lead rubbed some copper. Another was harder to locate, for it seemed to be within the magnet itself. Isolating the shorted parts of the apparatus from the measurement wires and current path solved the problem.

Burn-outs are a self explanatory problem that were only solved by experience and careful planning. Erring on the conservative side and doing more experiments that were incremental rather than attempting quick progress halted the problems. One particularly bad event happened when, in the course of being "extra careful" the power was switched to the pulse level, rather than off, at the end of a test. In the few moments that passed before realization of the error the burnout occurred. One burnout was expected as the pulses went up near 200 A at the 20 K level. Trying to get a quench in the inner coil was finally too much for the outer coil and it melted. Due to the design of the coil, with a lower grade conductor on the outside, burn-outs were always visible and on the outer layer. For one burnout an additional layer of conductor was removed to avoid the possibility that the heat of the burnout may have damaged the layers just below, creating the potential for additional problems. BSCCO is sensitive to heat and bending, and there is always a weakest spot on the conductor for the next problem to occur.

4.2 Simulation

4.2.1 Code

The diffusion equation is well discussed in most texts on numerical methods. It is not difficult to deal with in its basic form and lends itself to a multitude of methods. In an effort to be as accurate as possible, yet not knowing enough about the setup, too much complexity and not enough preparation was done before beginning to write the code. Benjamin Haid had previously worked on a similar project involving superconducting experimentation and computer simulation and his efforts seemed a good starting point for computer code. [4] In some ways this was the case, but by the time the code was complete it had been rewritten several times and starting from scratch might have been easier.

Written in ANSI C, the code is structured to be modular and readable. As can be seen in Appendix A, the beginning of the code contains definitions to make the code parametrically driven. The main body of the program outlines the procedure.

- Initialize output files
- Initialize materials matrix
- Initialize temperature
- Create property tables
- Begin time loop
- Make post-processing file
- Final output

The bulk of the code is in the time loop section where the algorithm is implemented and everything is processed in the explicit sequential time steps used by forward time methods, such as Lax.

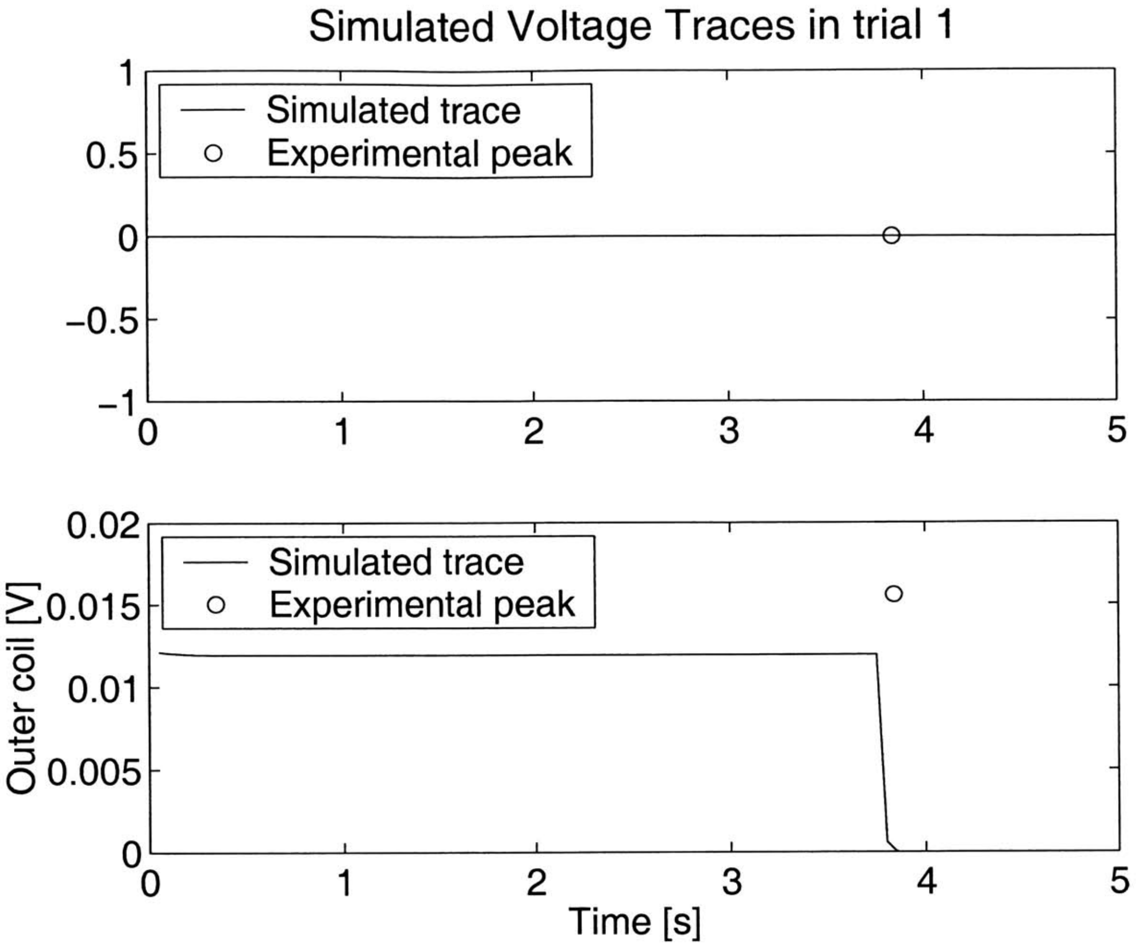

Figure 4-8: Simulated voltage traces for trial 1. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2. The open circle represents the peak voltage recorded in the experimental trials.

Through each iteration the code has become simpler and has been more flexible. Many of the initial complications remain, but their functionality removed, so they may be re-implemented if additional accuracy is needed. The number of nodes has decreased from an initial 1486 to 240. Not knowing much about the structure of the magnet initially, the code ignored some simplifications and made some wrong ones. Such errors have been fixed along the way and now the code runs, predicting reasonable voltages for some of the cases.

The simulation results for the trials are found in Figures 4-8, 4-9, 4-10, 4-11, 4-12, 4-13. With increased contact resistance the figures demonstrate the rising temperatures and a similar peaking profile to the experimental values. Further adjustment

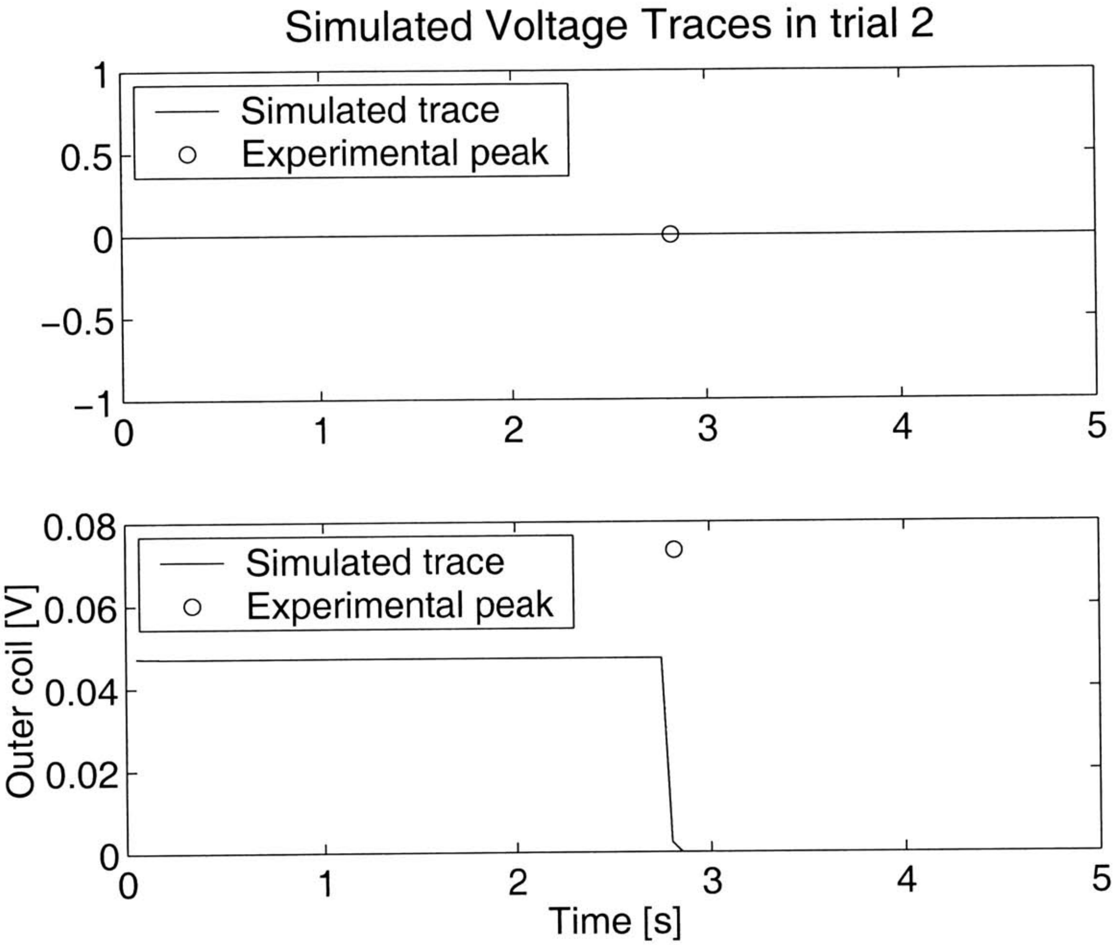

Figure 4-9: Simulated voltage traces for trial 2. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2. The open circle represents the peak voltage recorded in the experimental trials.

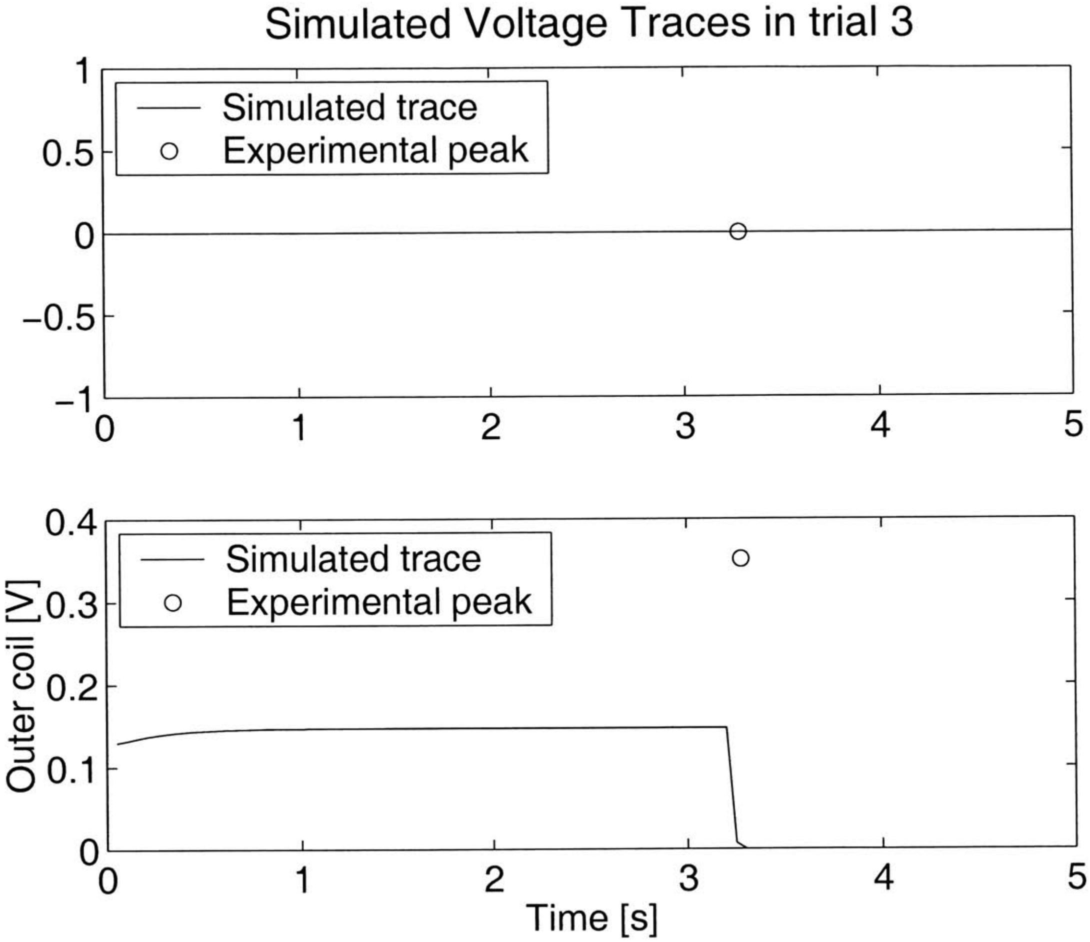

Figure 4-10: Simulated voltage traces for trial 3. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2. The dashed lines indicate the peak voltage observed in the experimental trials. The open circle represents the peak voltage recorded in the experimental trials.

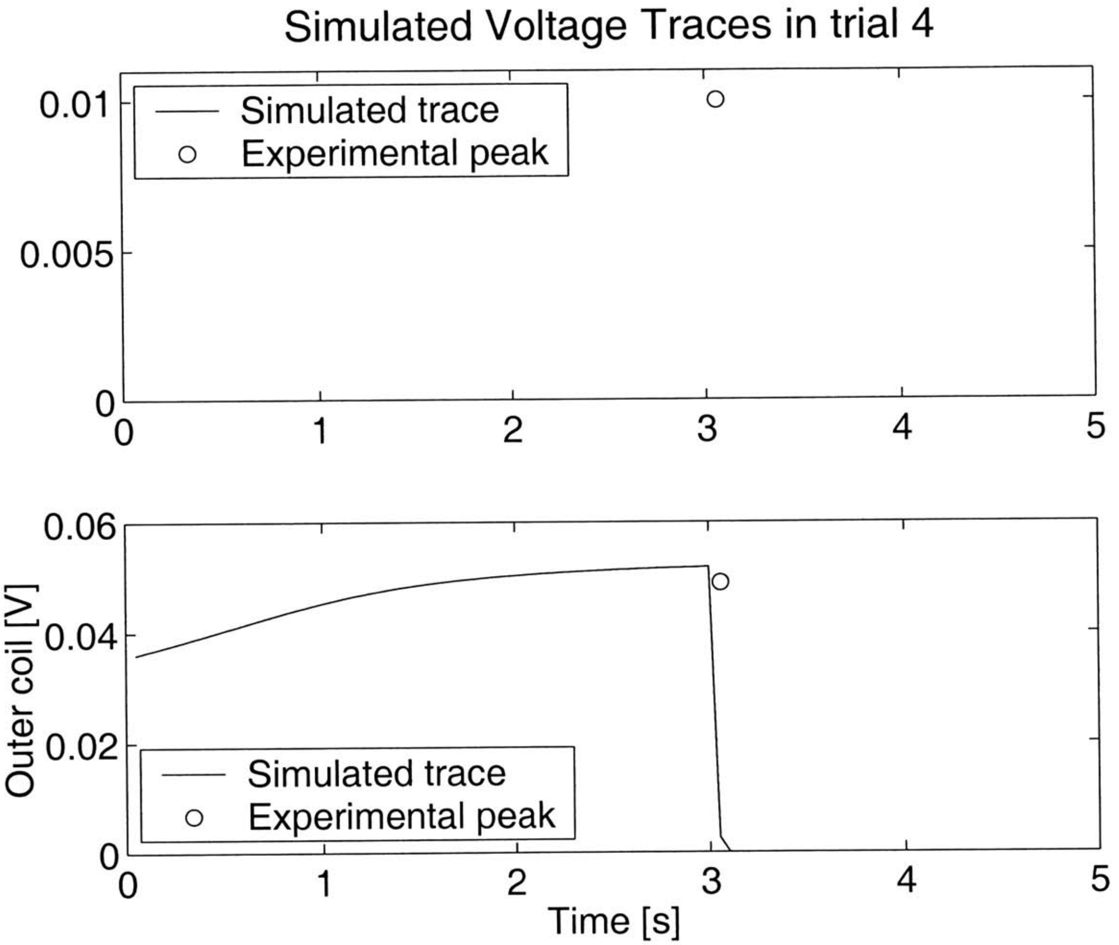

Figure 4-11: Simulated voltage traces for trial 4. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2. The open circle represents the peak voltage recorded in the experimental trials.

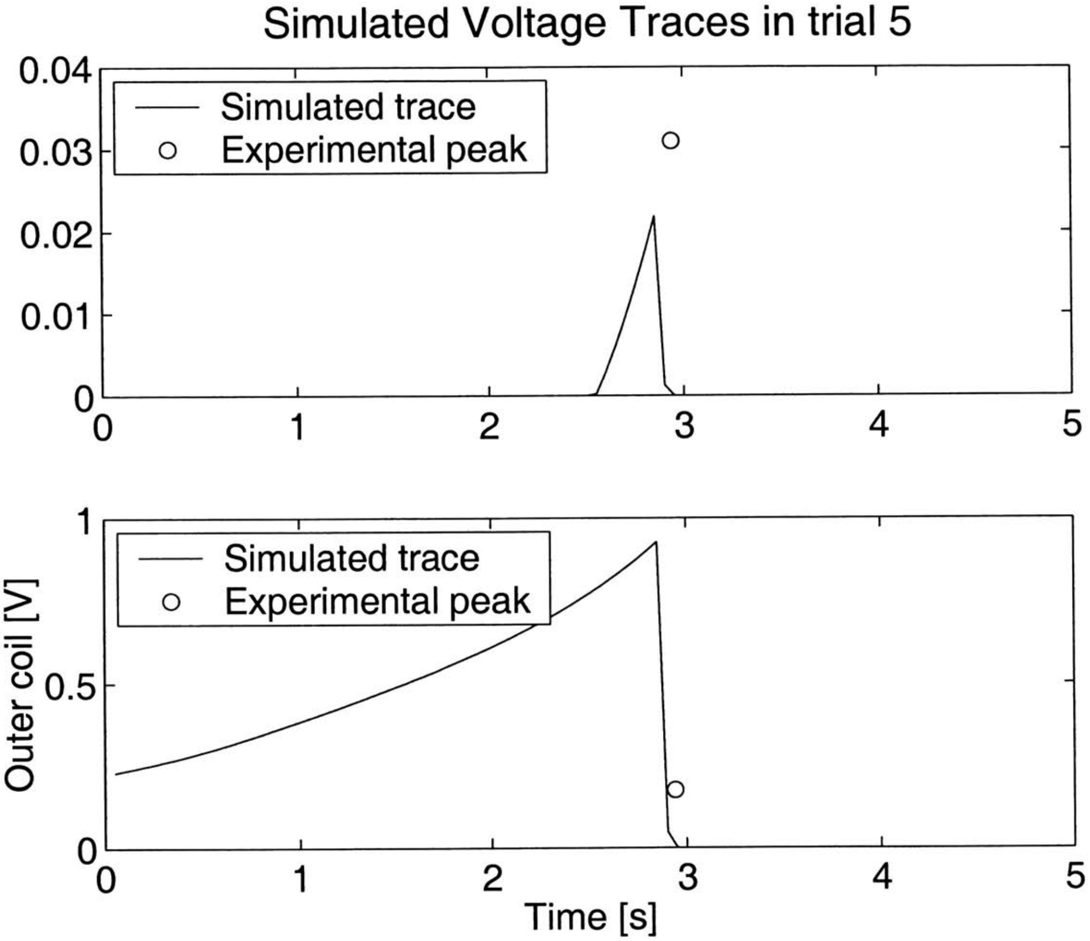

Figure 4-12: Simulated voltage traces for trial 5. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2.

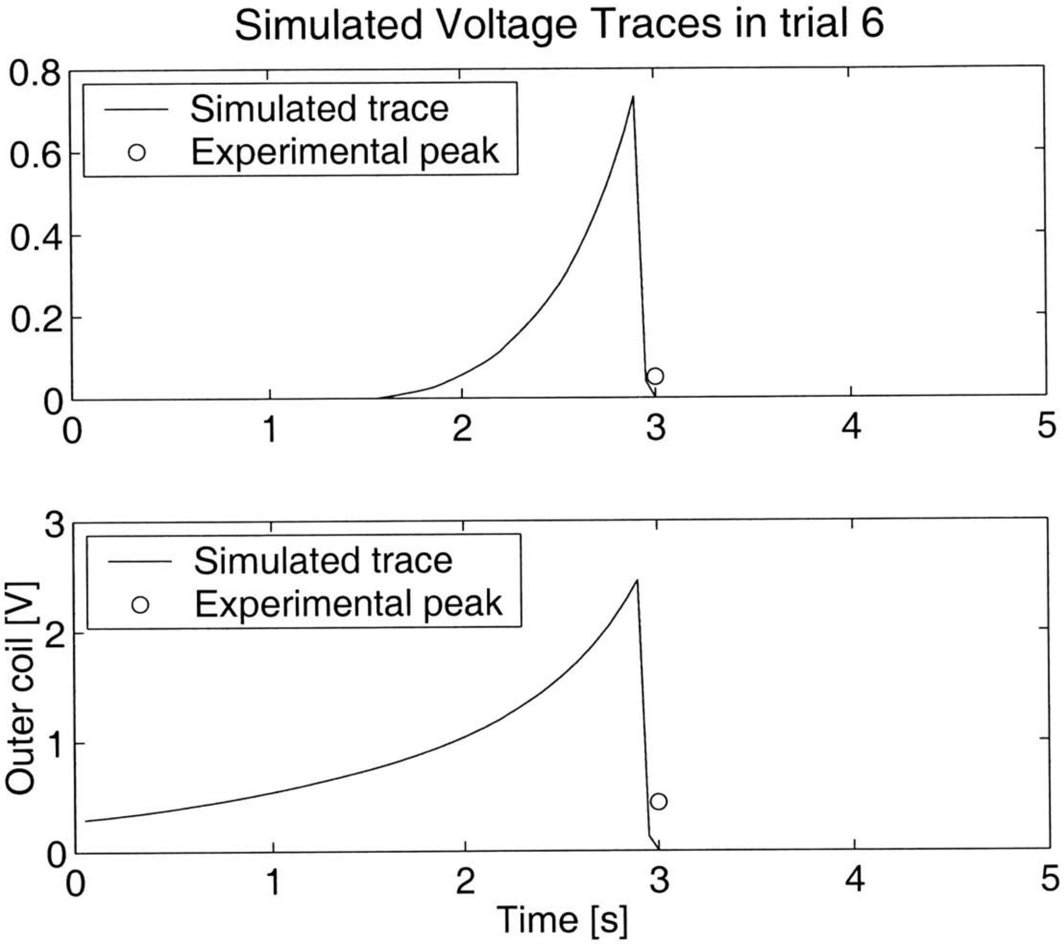

Figure 4-13: Simulated voltage traces for trial 6. The horizontal contact resistance used in the simulation was 12000. Maximum voltage comparisons are found in Table 4.2. The open circle represents the peak voltage recorded in the experimental trials.

Table 4.2: Maximum Voltage Comparison
Experimental vs. Simulated with R_H =12000

	Outer Coil		Inner Coil	
Trial	Simulated [V]	Experiment [V]	Simulated [V]	Experiment [V]
1	0.012	0.016	0	0
2	0.047	0.073	0	0
3	0.147	0.352	0	0
4	0.052	0.049	0	0.01
5	0.928	0.176	0.022	0.03
6	2.460	0.440	0.732	0.05

of parameters is not advisable to improve correlation since the simulation under predicts the 20 K cases and over estimates the voltages at 39 K. Table 4.2 compares the maximum voltages in the trials with the experimental data and shows that for the low current trials, relative to the critical values, the voltages are close. Although trial 5 shows similar voltages between experimental and simulated, the experimental profile was flat, indicating a current above critical in the inner coil. In the simulation the peak is indicative of conduction from the rapidly heating outer coil tainting the inner coil's current carrying capacity towards the end of the pulse.

The temperatures in trial 6, shown in Figure 4-14, show the expected increase of temperature due to joule heating in the conductor faster than it can be thermally conducted away. This observed temperature profile can only be obtained with large contact resistance. With smaller values of resistance thermal conduction is fast enough to set up a steady state with the heat generation and is also observable as a steady voltage level. Figure4-14 shows the simulated temperatures of trial 6, with the anticipated profile, where heat is generated faster than it can be conducted away. This profile can be created by increasing the horizontal contact resistance beyond the values initially used, which yielded nearly flat profiles. Adjusting the contact resistance to obtain the correct profile shape as seen in Figure 4-15 increased the voltages obtained in the simulation, as shown in Figure 4-16, and required re-formulating the critical current functions. The information considered and the experimental data's relation to the critical current functions used in the experiment are shown in Figure

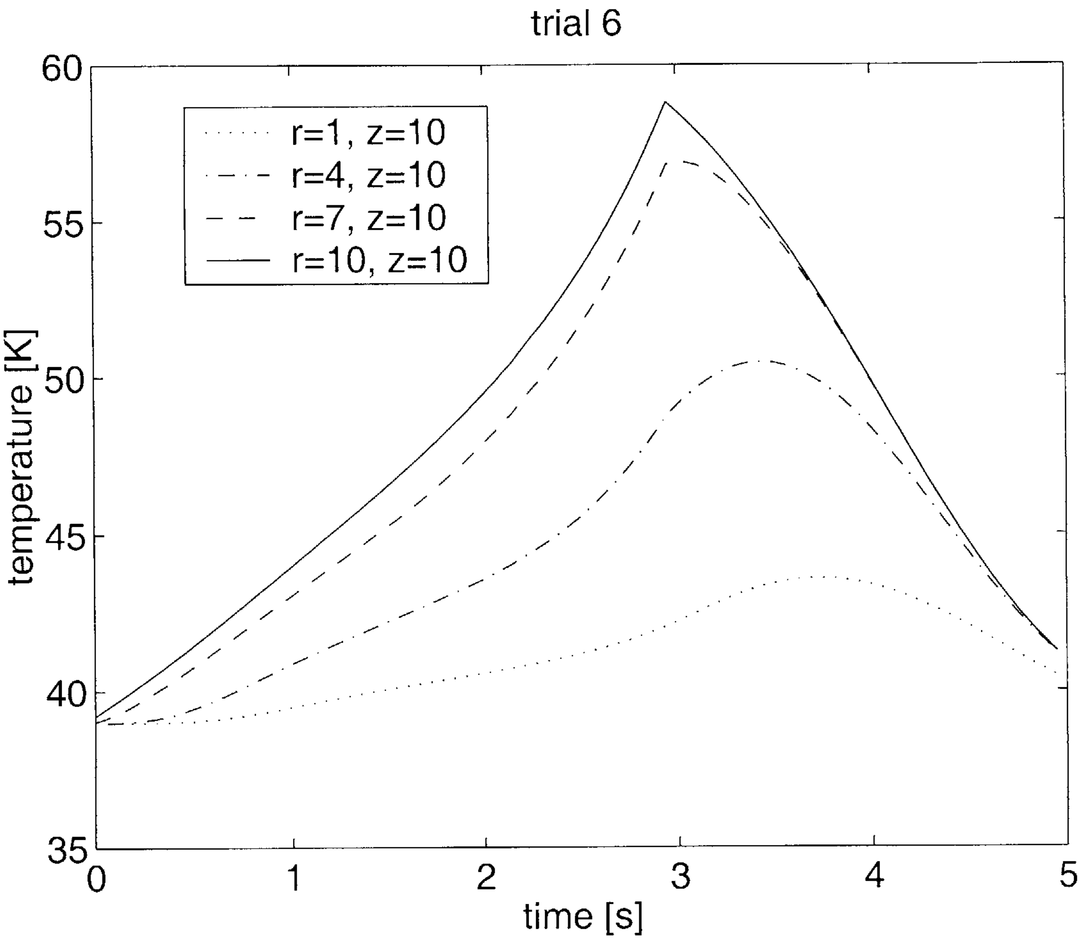

Figure 4-14: Temperatures in trial 6 at 4 locations along the centerline.

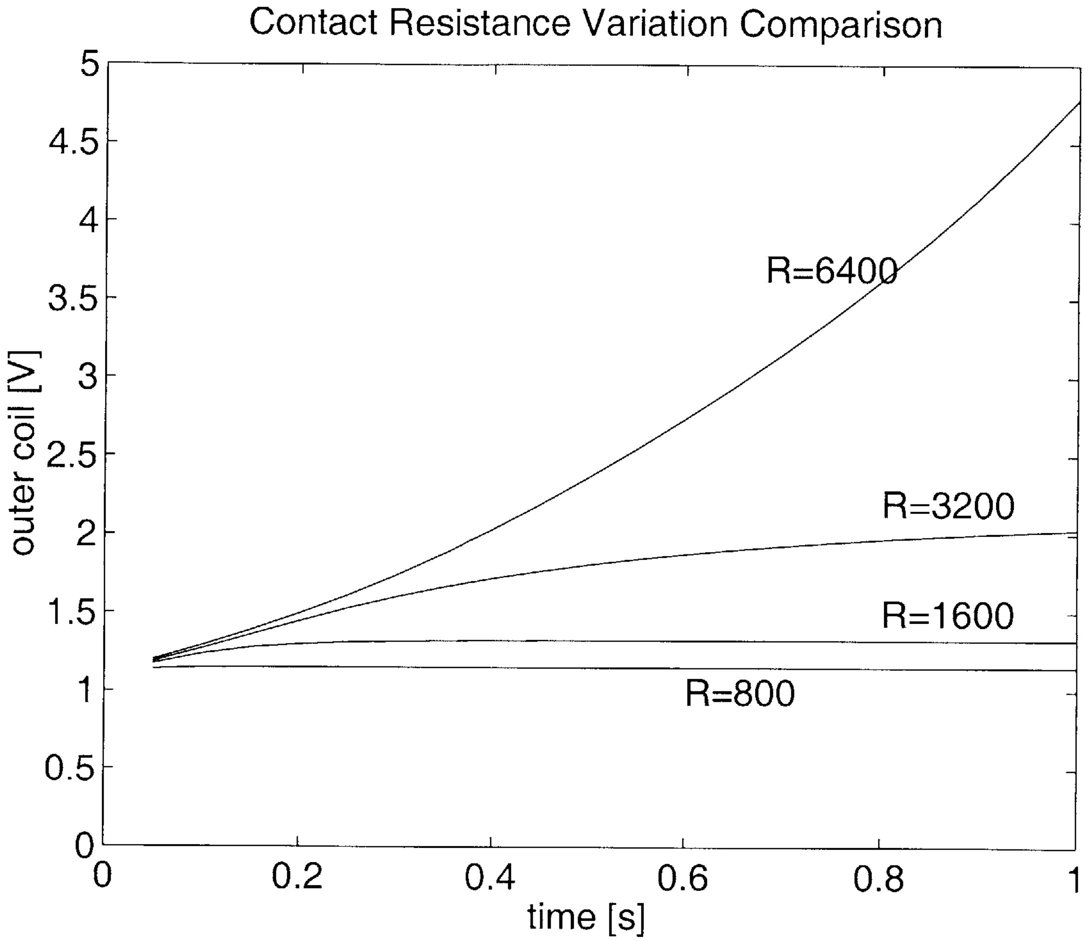

Figure 4-15: Change in voltage profile with increasing contact resistance. Graphs done with the original simulation parameters, including linear profiles for critical current, with I_c 160 A and 119 A for the inner and outer coils respectively at 20 K. The simulated Critical temperature is the same , at T_c= 89.5 K.

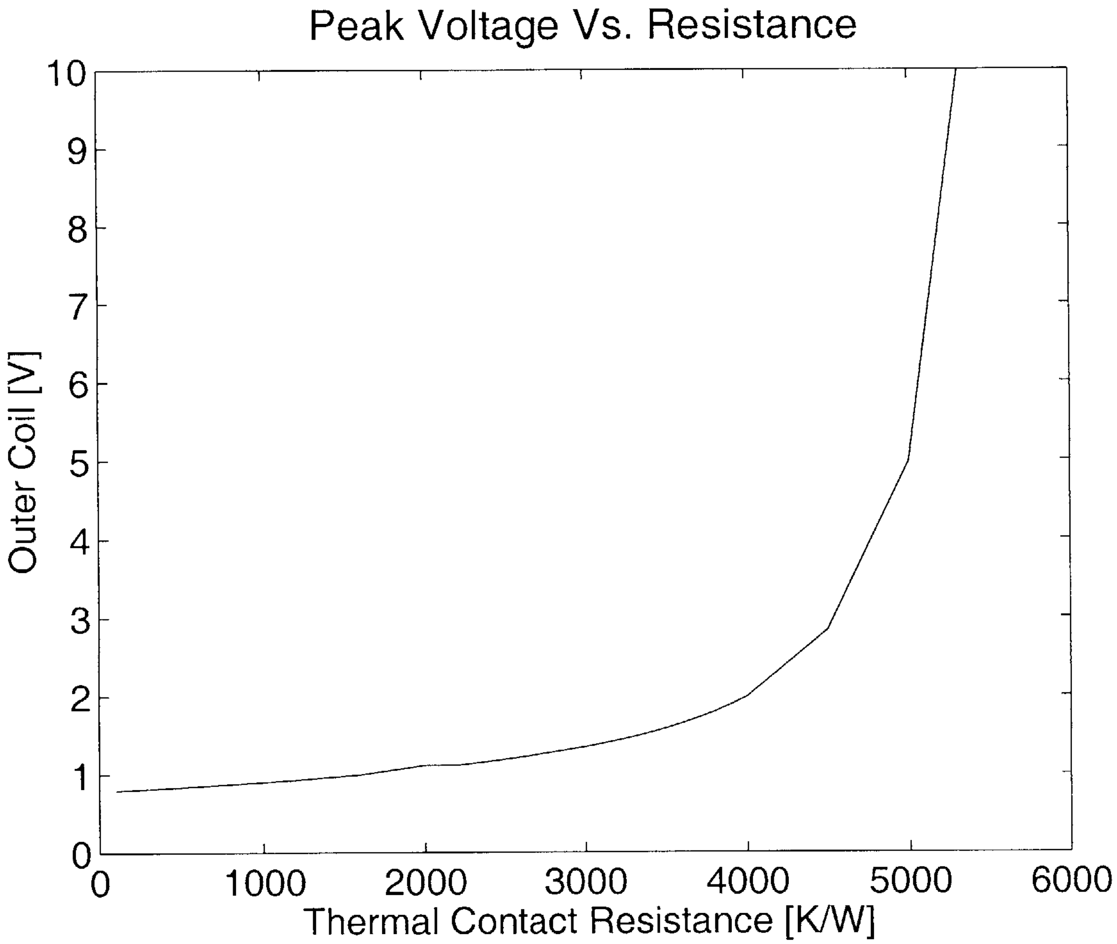

Figure 4-16: Peak voltage in outer coil as a function of horizontal contact resistance and using the original linear profiles for critical current, with I_c 160 A and 119 A for the inner and outer coils respectively at 20 K. T_c= 89.5 K as before.

4-17. To improve the accuracy of the simulation beyond this point, new information is required. The errors at the two temperature extremes are oriented so that they are mutually exclusive under the present modeling. Voltages at the cold end of the spectrum cannot be raised at the same time as lowering the high temperature voltages. There may be something novel going on thermally within the coil that has not yet been explained.

4.3 Further Considerations

The simulation obviously demonstrates that some of the assumptions being simulated are invalid. The one bright spot for the simulation is it closely predicts the voltage for the low current trials. This is not surprising the parameters of the simulation were adjusted for this purpose. Three items are necessary to accurately re-model the situation.

- The nature of the contact resistance within the magnet
- The nature of the transition between coils
- More time

Other factors, some of which were brought up elsewhere, could be solved with more time. The use of the Lax scheme, rather than the FTCS, has opened up the boundary condition options. Until recently only adiabatic walls were stable enough to complete a simulation. With the boundaries added, the importance of the contact resistance between adjacent materials has come to the forefront. Increasing the contact resistance has ironically made the FTCS scheme more stable because the nature of flux method creates a pseudo-second-order method that realizes its mathematical predicted stability problems at boundaries where the second order symmetry is broken.

In this case, contact resistance between materials seems to be the dominating factor in the heat transfer. With 3 layers of insulator between conductors and the coil dry wound there is poor contact between layers. An explanation for the discrepancy is on

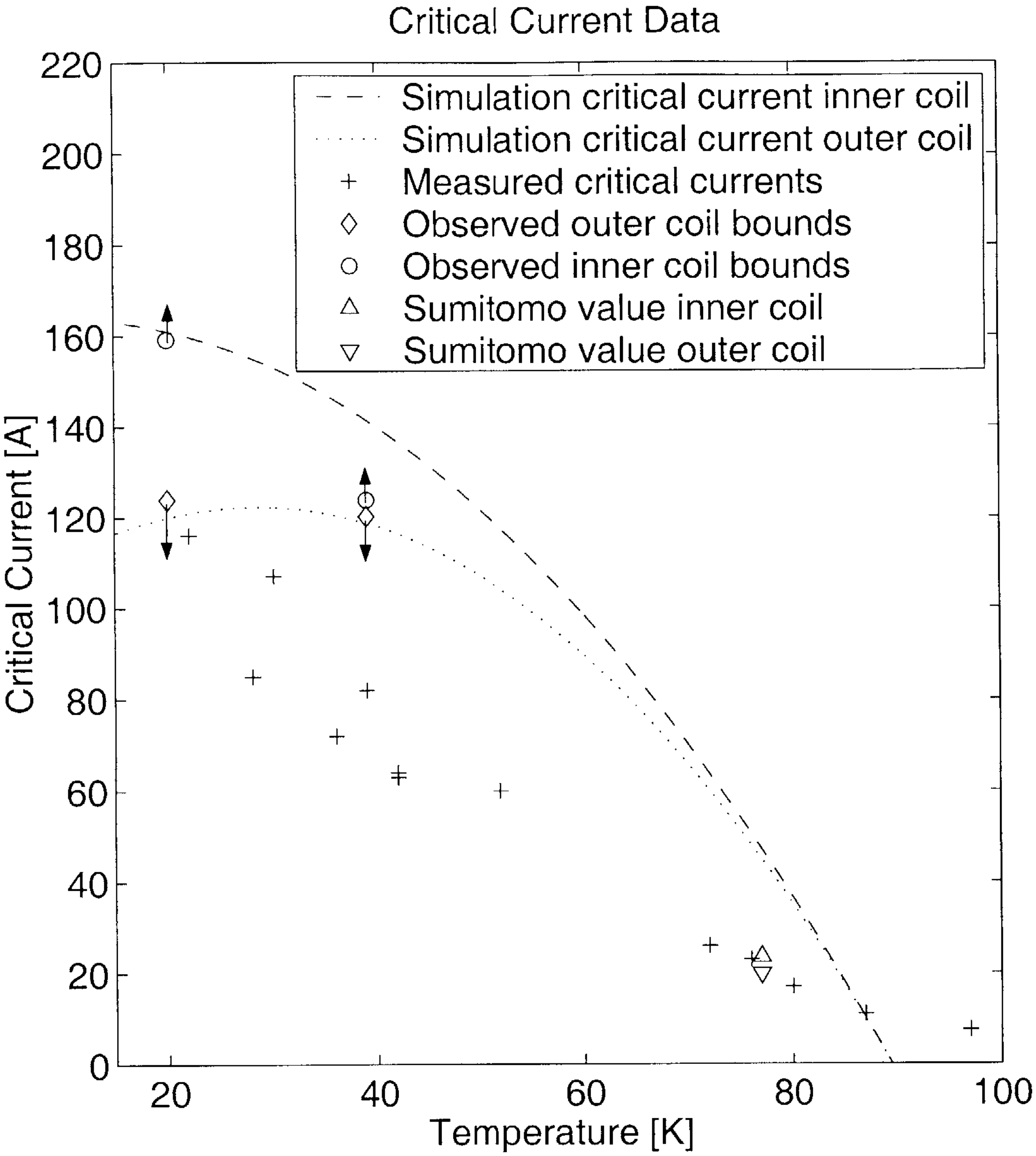

Figure 4-17: Experimental Data and it's relation to the critical current parameters used in the simulation.
The (+) were measured using the test coil to confirm the reported parameters given in the specifications and were very rough. They were measured based on the outer coil being at 1 μV/cm with both coils being ramped at 0.5 A/s.
The Sumitomo values (△▽) were provided conductor specifications that came with the magnet.
The bounds (◇○) are based on observed voltages, or lack thereof, in pulsing experiments and represented the limit to which the critical currents could be adjusted without defying observation.

the first order of business before continuing with further experimentation. Chyu and Oberly, who studied the influence of thermal resistance on normal zone propagation in metal sheathed HTS tape, concluded by saying that contact resistance was insignificant, unless that resistance was comparable in both transverse and longitudinal directions, which would decrease NZP and increase local heating. [1] The orientation of the turns in the coil makes conduction in the z direction almost non-existent. There is no pressure in that direction and the only connection is by a 25μm layer of Kapton 0.08 mm long. In the original simulation, heat transfer between layers was restricted in the code to along the Kapton. With the ensuing simplification and reduction of nodes it was more convenient to create a "vertical contact resistance" that was independent of the horizontal to imitate the resistance to flow in the vertical direction. If the resistance in the other direction were not so significant, it could change the nature of the simulation. For the number of assumptions required, a simplified analytical approach might be a best first step.

Having two separate coils and material properties complicated the simulation, but not nearly as much as it hindered direct observation of the experiment. Other than measuring the voltage drop across the inner coil there was no way to measure any of the properties. Thermocouples could not be inserted to measure temperature within the inner layer, next to the coil form. Those inner layers may not be well anchored thermally to the wall. Between layers there are double stick pads to maintain tension in the conductor winding . When repairing burn-outs we relied on these pads to prevent the coil from unraveling. At the interface between coils there does not need to be such a pad to maintain contact or provide tension. The interface between coils may have a much higher contact resistance than the rest of the coil. Such a possibility would explain the shape of the voltage pulse as heat was generated. The inner coil may be thermally anchored to the coil form and the heat is maintained in the outer coil. The small observed voltages would be from the outer portion of the inner coil, heated slightly by the outer coil, but maintaining enough conduction to prevent quenching. It is also possible that the contact resistance is a function of radial distance, and that the closer the layers are to the coilform, the more pressure there is creating thermal contact. All such ideas are only speculation and additional investigation would be required to confirm or rule them out.

The limitations of this experiment preclude making any real predictions. The anticipated cases of quench and recovery have been observed. The lower, 20 K, temperature is very stable and capable of high currents. Quenches at 20 K generally recover quickly and the material properties aid in this process. Repeated experimentation at higher temperatures required longer to reach equilibrium and were easier to quench. The simulation algorithm is valid, but the implementation needs to be streamlined to work with modified assumptions involving heat transfer between layers. The significance of contact resistance has been observed, but its effect has not been quantified. This needs to become a part of any simulation along with an improved thermal understanding of the coil and a verified critical model of the conductor.

Chapter 5

Conclusions

Beginning with a set of assumptions regarding joule heating of a superconducting magnet, three scenarios were postulated. If quenching did occur, heating would result. With a stepped down voltage the quench could continue to grow, it could shrink due to transient conduction, or it could be superconducting at the reduced current despite any heating during the pulse. All of these events have been observed and the transitions noted as smooth. Reduction in the level of current will prevent damage to the conductor, but thermal recovery is most effective in thge superconducting state. To model the experimental observations a C program numerically solved the transient thermal diffusion equation. Included were the variations of the material properties with temperature, and with position in the magnet. That program can accurately predict the voltage of the coil during a pulse, but does not accurately predict the observed temperature growth, nor the differences between operation at 20 K and 39 K. Errors in modeling of the conduction between layers and the effect of contact resistance are the assumed cause.

Bibliography

[1] M. K. Chyu and C. E. Oberly. Influence of operating temperature and contact thermal resistance on normal zone propagation in a metal-sheathed high-tc superconductor tape. *Cryogenics*, 32(5):519, 1992.

[2] L. Dresner. Stability and protection of Ag/BSCCO magnets operated in the 20-40 K range. *Cryogenics*, 33:900, 1993.

[3] Alejandro L. Garcia. *Numerical Methods for Physics.* Prentice Hall, Englewood Cliffs, NJ, 1994.

[4] Benjamin J. Haid. Two-dimensional quench propogation model for a three-dimensional "High-Temperature" superconducting coil. Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, June 1998.

[5] R.M. Hazen. *The Breakthrough - the race for the superconductor.* Summit Books, 1988.

[6] J. P. Holman. *Heat Transfer.* McGraw-Hill, Inc., 1997.

[7] Yukikazu Iwasa. *Case Studies in Superconducting Magnets.* Plenum Press, New York and London, 1994.

[8] Yukikazu Iwasa. Mit-ornl experiment proposal. Presentation on Phase I (this thesis work) and Phase II of of a thermal recovery experiment in collaboration with Oakridge National Laboratory, 1999.

[9] Hunwook Lim. *Normal Zone Propagation in Hight-Temperature Superconducting Tape-Wound Coils*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, February 1996.

[10] K. Noto, H. Fujishiro, K. Michishita, M. Matsukawa, F. Tatezaki, and Y. Kubo. Thermal conductivity of AG-doped Bi-2212 superconducting materials prepared by the floating zone method. *Cryogenics*, 34, 1994.

[11] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipies in C*. Cambridge University Press, Cambridge, United Kingdom, 1999.

[12] Jr Robert D. Pillsbury. *SOLDESIGN User's Manual*. Plasma Fusion Center Massachusetts Institute of Technology, Cambridge, MA, February 1991.

[13] R. B. Scott. *Cryogenic Engineering*. Met-Chem Research Inc., 1988.

[14] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Springer-Verlag, New York, 1995.

[15] Y. Touloukian. *Thermophysical Properties of Matter*, volume 1-12. IFI/Plenum, New York, 1970.

[16] K. Sato Y. Iwasa, R.H. Bellis. Magnetoresistivity of silver over temperature range 4.2-159 K. *Cryogenics*, 33:836, 1993.

Appendix A

Simulation Code

```
/* prog.c                  */
/* temp simulation program  */
/*                          */
/* by                       */
/* Karl Kowallis            */
/* December 1999            */
/*      v11                 */
/*                          */
/* for use in MSME thesis work */
/* under the direction of   */
/* Yuki Iwasa               */

#include <stdio.h>

#if defined(__STDC__) && !defined(__HIGHC__)
#include <stdlib.h>
#endif

#include <math.h>


/* definitions section */

#define TOTALTIME       5.0      /* s */
```

```
#define PULSETIME       3.0     /* s */
#define IOP           27.71   /* A */  /* operating current Amps      */
#define IPULSE          130.615 /* A */  /* total current Pulse  Amps  */
#define TEMPOP          39.     /* K */
#define TINITIAL        39.     /* K */
#define REFFILE         "trial 6"

#define DT            1e-4  /* s */
#define TIMESTEPS       (int)(TOTALTIME/DT)
#define SAMPLERATE      500

#define BOP           (0.2 * IOP/100) /* T */ /* field at operating current */
#define BPULSE          (BOP*(IPULSE/IOP))      /* pulse current Field          */
#define CON_W           0.35      /* cm */        /* larger  dimension  */
#define CON_H           0.023     /* cm */        /* smaller dimension  */
#define INSULTHICKNESS 0.0025    /* cm */
#define RADIUS          5.29      /* cm */
#define PI            3.14159
#define CIRCUMFERENCE  (2*PI*RADIUS)
#define COIL_H          0.2       /* cm */
#define COIL_W          0.2       /* cm */
#define INN_LAYERS      8
#define OUT_LAYERS      4
#define LAYERS          (INN_LAYERS+OUT_LAYERS)
#define TURNSpLAYER     20

#define IC1A            161.0         /* A */  /* critical current at 20 K    */
#define IC1B            140.0         /* A */  /* critical current at 40 K    */

#define TC3             89.5          /* K */  /* critical temperature        */

#define IC2A            120.0         /* A */  /* critical current at 20 K    */
#define IC2B            118.0         /* A */  /* critical current at 40 K    */

#define SILVERFRACTION 0.735                    /* volumetric fraction silver  */
#define CON_AREA        (CON_H*CON_W*SILVERFRACTION) /* AG cross sectional area*/
```

```
#define CONTACT_H       12000  /* [K/W] */ /* horizontal contact resistance  */
#define CONTACT_V       100000 /* [K/W] */ /* vertical   contact resistance  */
#define FORM_FACTOR     1      /* ratio of energy allowed to enter coilform  */
#define CONDUCTOR_INN  1
#define CONDUCTOR_OUT  2

/* fit paramaters for cryocooler power vs. temp */
#define P1            -0.000217715011817744
#define P2             0.239874719044138
#define P3             (-P1*TEMPOP*TEMPOP-P2*TEMPOP)

#define TTOT           8000            /* points for property interpolation */
#define FACTOR          3              /* additional data points for ctab   */
#define ZTOTAL          TURNSpLAYER
#define RTOTAL          LAYERS
#define PERIM          (RTOTAL+RTOTAL+ZTOTAL)
#define ADIABATIC       0
#define COILSPACE       0
#define COILSFACTOR     0.9


/* Property function prototypes */

float kcalc(float T, int mtrl, float b);
float rhocalc(float T, float b);
float ccalc(float T, int mtrl);

/* Procedural Prototypes */

void FinalOutput(double T[RTOTAL][ZTOTAL], char outputV[], char outputT[]);
void WriteTempData(char name[] ,double T[RTOTAL][ZTOTAL]);
void WriteVoltData(char name[], float volts[LAYERS][TURNSpLAYER]);
void TimeLoop(int  mtrl[RTOTAL][ZTOTAL],
        float  ktab[TTOT][3], float ctab[TTOT*FACTOR], char outputT[],
        char outputV[], double T[RTOTAL][ZTOTAL], double boundaryT[PERIM]);
```

```
void CreateTables(float ktab[TTOT][3], float ctab[TTOT*FACTOR], float field);
void InitializeFile(char name[]);
void MakeInfofile(char *info,char *outputV, char *outputT);
void InitializeMaterials(int mtrl[RTOTAL][ZTOTAL]);
void InitializeTemp(double T[RTOTAL][ZTOTAL], double boundaryT[PERIM]);


/***  Global variables  ***/

float COEFA2;
float COEFA1;
float COEFB2;
float COEFB1;
float COEFC2;
float COEFC1;




main()
{
  float ktab[TTOT][3], ctab[TTOT*FACTOR];
  char *outputV="volt.dat", *outputT="temp.dat", *info="siminfo.m";
  int mtrl[RTOTAL][ZTOTAL];
  double T[RTOTAL][ZTOTAL], boundaryT[PERIM];

  COEFA2=-1.0/20*(-IC2A*TC3+40*IC2A+IC2B*TC3-20.0*IC2B)/
    (TC3-20.)/(TC3-40.0);
  COEFA1=-1.0/20*(-IC1A*TC3+40*IC1A+IC1B*TC3-20.0*IC1B)/
    (TC3-20.)/(TC3-40.0);
  COEFB2=1.0/20*(-IC2A*TC3*TC3+1600*IC2A+IC2B*TC3*TC3-400*IC2B)/
    (TC3-20)/(TC3-40.0);
  COEFB1=1.0/20*( 1600*IC1A-IC1A*TC3*TC3+IC1B*TC3*TC3-400*IC1B)/
    (TC3-20)/(TC3-40.0);
  COEFC2=-TC3*(-2*IC2A*TC3+80*IC2A+IC2B*TC3-20.0*IC2B)/
    (TC3-20.0)/(TC3-40.0);
  COEFC1=-TC3*(-2*IC1A*TC3+80*IC1A+IC1B*TC3-20.0*IC1B)/
```

```
    (TC3-20.0)/(TC3-40.0);

  printf("Begin main()\n"); fflush(stdout);


  printf("InitializeFile(outputV);\n"); fflush(stdout);
  InitializeFile(outputV);
  printf("InitializeFile(outputT);\n"); fflush(stdout);
  InitializeFile(outputT);
  InitializeMaterials(mtrl);
  InitializeTemp(T,boundaryT);
  printf(" InitializeTemp(T);  --Done--\n"); fflush(stdout);
  CreateTables(ktab, ctab, BPULSE);
  TimeLoop(mtrl, ktab, ctab, outputT, outputV, T, boundaryT);
  printf(" TimeLoop --Done--\n"); fflush(stdout);
  printf("Begin MakeInfoFile\n"); fflush(stdout);
  MakeInfofile(info,outputV,outputT);
  printf(" MakeInfoFile -- Done --\n"); fflush(stdout);
  printf("Begin FinalOutput()"); fflush(stdout);
  FinalOutput(T, outputV, outputT);
  printf(" FinalOutput --Done--\n"); fflush(stdout);
  return(0);
}

/**********************************************************************/
/* function InitializeTemp ***********************************************/
/**********************************************************************/
void InitializeTemp(double T[RTOTAL][ZTOTAL],double boundaryT[PERIM])
{
  int r,z;

  printf("Begin InitializeTemp\n");  fflush(stdout);
  for (r=0; r< RTOTAL; r++)
    for (z=0; z< ZTOTAL; z++)
      T[r][z]=TINITIAL;
  for (z=0; z< PERIM; z++)
    boundaryT[z]=(TEMPOP);
  for (z=0; z< ZTOTAL; z++)
   {
     for (r=0; r< RTOTAL; r++)
      {
        printf("%2.1f ",T[r][z]); fflush(stdout);
      }
     printf(" z=%d\n",z); fflush(stdout);
   }

}   /*** end InitializeTemp ***/

/**********************************************************************/
/* function MakeInfofile ************************************************/
/**********************************************************************/
void MakeInfofile(char *info, char *outputV, char *outputT)
{
  FILE *ofp;

  ofp=fopen(info,"w");
  fprintf(ofp,"");
  fclose(ofp);

  fopen(info,"a");
  fprintf(ofp,"RefFile='%s';\n",REFFILE);
  fprintf(ofp,"Tempfile='%s';\n",outputT);
  fprintf(ofp,"Voltfile='%s';\n",outputV);
  fprintf(ofp,"LAYERS=%d;\n",LAYERS);
  fprintf(ofp,"TURNSpLAYER=%d;\n",TURNSpLAYER);
  fprintf(ofp,"TOTALTIME=%d;\n",TOTALTIME);
  fprintf(ofp,"SAMPLERATE=%d;\n",SAMPLERATE);
  fprintf(ofp,"ZTOTAL=%d;\n",ZTOTAL);
  fprintf(ofp,"RTOTAL=%d;\n",RTOTAL);
  fprintf(ofp,"DT=%f;\n",DT);
  fprintf(ofp,"TIMESTEPS=%d;\n",TIMESTEPS);
  fprintf(ofp,"INN_LAYERS=%d;\n",INN_LAYERS);
```

```
    fprintf(ofp,"OUT_LAYERS=%d;\n",OUT_LAYERS);
    fprintf(ofp,"RADIUS=%f;\n",RADIUS);
    fprintf(ofp,"CIRCUMFERENCE=%f;\n",CIRCUMFERENCE);
    fprintf(ofp,"COIL_H=%f;\n",COIL_H );
    fprintf(ofp,"COIL_W=%f;\n",COIL_W );
    fprintf(ofp,"CON_AREA=%f;\n",CON_AREA);
    fprintf(ofp,"IC1A=%f;\n",IC1A);
    fprintf(ofp,"IC1B=%f;\n",IC1B);
    fprintf(ofp,"IC2A=%f;\n",IC2A);
    fprintf(ofp,"IC2B=%f;\n",IC2B);
    fprintf(ofp,"TC3=%f;\n",TC3);
    fprintf(ofp,"COEFA1=%f;\n",COEFA1);
    fprintf(ofp,"COEFB1=%f;\n",COEFB1);
    fprintf(ofp,"COEFC1=%f;\n",COEFC1);
    fprintf(ofp,"COEFA2=%f;\n",COEFA2);
    fprintf(ofp,"COEFB2=%f;\n",COEFB2);
    fprintf(ofp,"COEFC2=%f;\n",COEFC2);
    fprintf(ofp,"BOP =%f;\n",BOP );
    fprintf(ofp,"BPULSE=%f;\n",BPULSE);
    fprintf(ofp,"IPULSE=%f;\n",IPULSE);
    fprintf(ofp,"IOP =%f;\n",IOP );
    fprintf(ofp,"CON_W=%f;\n",CON_W );
    fprintf(ofp,"CON_H=%f;\n", CON_H);
    fprintf(ofp,"INSULTHICKNESS=%f;\n",INSULTHICKNESS);
    fprintf(ofp,"RADIUS=%f;\n",RADIUS);

    fprintf(ofp,"disp('Simulation currents were %3.f A ",
            IPULSE);
    fprintf(ofp,"and %2.1f A with a %1.2f s pulse.')",
            IOP,PULSETIME);
    fclose(ofp);

}   /*** end MakeInfofile ***/

/**************************************************************/
/* function FinalOutput ******************************************/
```

```
/**********************************************************************/
void FinalOutput(double T[RTOTAL][ZTOTAL], char outputV[], char outputT[])
{
  int r=0 ,z=0 ,rmax=0 ,zmax=0;
  float Tmax=0;

  printf("Begin FinalOutput\n"); fflush(stdout);

  for (r=0; r< RTOTAL; r++)
    for (z=0; z< ZTOTAL; z++)
      if (T[r][z]>Tmax)
        {
          Tmax=T[r][z];
          rmax=r;
          zmax=z;
        }
  if (Tmax <= 500.0)
    {
      printf("----------------------------------\n");
      printf("Simulation completed successfully.\n");
      printf("----------------------------------\n");
      printf("The Maximum final temperature was %f at node r=%d z=%d",
             Tmax,rmax,zmax);
      printf("\nOutput written to:\n");
      printf("\ttemperature:\t%s\n",outputT);
      printf("\tvoltage:\t%s\n\n",outputV);
      printf("Files are written with RTOTAL=%d columns\n",RTOTAL);
      printf("Files are written with ZTOTAL=%d rows\n",ZTOTAL);
      printf("Data was collected every %d steps for a total of %d sets\n\n"
             ,SAMPLERATE, (int)(TIMESTEPS/SAMPLERATE));
    }
}   /*** end FinalOutput ***/

/**********************************************************************/
/* function WriteTempData ********************************************/
/**********************************************************************/
```

```
void WriteTempData(char name[] ,double T[RTOTAL][ZTOTAL])
{
  FILE *ofp;
  int r,z;


  ofp=fopen(name,"a");

  for (z=0; z<ZTOTAL; z++)
    {
      for (r=0; r<RTOTAL; r++)
        fprintf(ofp,"%3.2f ",T[r][z]);
      fprintf(ofp,"\n");
    }
  fclose(ofp);
}   /***end WriteTempData ***/

/**********************************************************************/
/* function WriteVoltData **********************************************/
/**********************************************************************/
void WriteVoltData(char name[], float volts[LAYERS][TURNSpLAYER])
{
  FILE *ofp;
  int r,z;

  ofp=fopen(name,"a");

  for (z=0; z<TURNSpLAYER; z++)
    {
      for (r=0; r<LAYERS; r++)
        fprintf(ofp,"%1.6e ",volts[r][z]);
      fprintf(ofp,"\n");
    }
  fclose(ofp);
}   /*** end WriteVoltData ***/
```

```
/**********************************************************************/
/* function TimeLoop **************************************************/
/**********************************************************************/
void TimeLoop(int mtrl[RTOTAL][ZTOTAL],
              float  ktab[TTOT][3], float ctab[TTOT*FACTOR],
              char outputT[], char outputV[],
              double T[RTOTAL][ZTOTAL], double boundaryT[PERIM])
{
  int time=0,r=0,z=0;
  int stop=0;
  float c, rho, Ic;
  float qin=0, qout=0, qcond=0, qrad=0, qnz=0;
  float qcooler=0, qtop=0, qbottom=0, qsink=0;
  float qovertop=0, qoverbottom=0;
  float invRbottom=1, invRright=1;
  /*  float rm=0; */
  float outtime;
  float current=IPULSE;
  float field=BPULSE;
  float Tavc=0;
  float q1[ZTOTAL], q2[ZTOTAL], *qright=NULL, *qleft=NULL, *floatptr=NULL;
  float volts[LAYERS][TURNSpLAYER];

  /* procedures used only in TimeLoop */

  float invrightcalc(double T[RTOTAL][ZTOTAL],
                float ktab[TTOT][3], int r, int z);
  float invbottomcalc(double T[RTOTAL][ZTOTAL],
                float ktab[TTOT][3], int r, int z);
  float conductorcalc(double Tlocal, double Tneighbor,
                      float ktab[TTOT][3]);   /* returns + if (Tl > Tn) */
  float cryocooler(double T);

  /* begin main body of function */

  printf("Begin TimeLoop\n"); fflush(stdout);
```

```
qleft=q1;
qright=q2;
printf(" time loop begins\n"); fflush(stdout);
for (time=1; time<=TIMESTEPS; ++time)
  {
    if (fabs(time*DT-PULSETIME)<=0.5*DT)
      {
        printf(" End Pulse; Time*DT== PULSETIME\n");  fflush(stdout);
        CreateTables(ktab, ctab, field=BOP);
        current=IOP;
      }
    if ((time%SAMPLERATE)==0)
      {
        outtime=(time/SAMPLERATE);
        printf(" timestep %d of %d at %f sec.\n",
                time, TIMESTEPS, time*DT ); fflush(stdout);
      }
    else
      outtime=0;

    /* calc cryocooler heat withdrawl */


    /* r=0 case initialization */
    r=0;
    qsink=0;
    for (z=0; z<ZTOTAL; ++z)
      {
        if (ADIABATIC)
          qleft[z]=0;  /* Adiabatic boundary condition  */
        else
          {            /* Fixed temp coilform boundary condition */
            invRright=invrightcalc(T,ktab,r+1,z);
            qleft[z]=FORM_FACTOR*invRright*(boundaryT[z]-T[r][z]);
            qsink+=-qleft[z];
          }
  }
/* Begin main r,z loop */

for (r=0; r<RTOTAL; ++r)
  {
    z=0;
    if (ADIABATIC)
      qtop=0;  /* Adiabatic boundary condition  */
    else
      {        /* Fixed temp coilform boundary condition */
        invRbottom=invbottomcalc(T, ktab, r, z+1);
        qtop=FORM_FACTOR*invRbottom*(boundaryT[ZTOTAL+r]-T[r][z]);
        qsink+=-qtop;
      }
    Tave=2*T[r][z+1];
    if (!r && !((RTOTAL-1)==r))
      Tave+=T[r+1][z]+T[r-1][z];
    else
      Tave+=2*T[(!r+(r==(RTOTAL-1))*(RTOTAL-2))][z];



    Tave=2*T[r+1][z];
    if (!z && !((ZTOTAL-1)==z))
      Tave+=T[r][z-1]+T[r][z+1];
    else
      Tave+=2*T[r][(!z+(z==(ZTOTAL-1))*(ZTOTAL-2))];

    for (z=0; z<ZTOTAL ; ++z)
      {
        if (z<ZTOTAL-1)
          {
            invRbottom=invbottomcalc(T, ktab, r,z);
            qbottom= (T[r][z]-T[r][z+1])*invRbottom ;
            /* calc qbottom */
  }
else /* bottom of coil */
  if (ADIABATIC)
    qbottom=0;  /* Adiabatic Boundary Condition */
  else
    {            /* Fixed temp coilform boundary condition */
      invRbottom=invbottomcalc(T, ktab, r, z-1);
      qbottom=FORM_FACTOR*invRbottom*
        (boundaryT[ZTOTAL+r]-T[r][z]);
      qsink+=qbottom;
    }

if (r<RTOTAL-1)
  {
    invRright= invrightcalc(T, ktab, r,z);
    if ((COILSPACE)&&(r==(INN_LAYERS-1)))
      qright[z]=COILSFACTOR*(T[r][z]-T[r+1][z])*invRright ;
    /* calc qright */
    else
      qright[z]=(T[r][z]-T[r+1][z])*invRright ;
    /* calc qright */
  }
else  /* right edge of coil  */
  if (ADIABATIC)
    qright[z]=0;  /* Adiabatic Boundary Condition */
  else
    {
      /*  Include radiation input along outer edge here     */
      /*  it should be calculated and included in qright[z]*/
      /*    qrad=0; */

      invRright=0;
      qright[z]=0;
    }

if (r && z)
```

```
    if ((r<(RTOTAL-1))&&(z<(ZTOTAL-1)))
      Tave=T[r+1][z]+T[r][z+1]+T[r-1][z]+T[r][z-1];
    else
      if (r==(RTOTAL-1))
        Tave=T[r][z-1]+
          2*T[r-1][z]+
          T[r][(z==(ZTOTAL-1))*(z-1)+(z<(ZTOTAL-1))*(z+1)];
      else
        Tave=T[r-1][z]+
          2*T[r][z-1]+
          T[(r==(RTOTAL-1))*(r-1)+(r<(RTOTAL-1))*(r+1)][z];
  else
    if (!r)
      if (z)
        Tave=2*T[r+1][z]+
          T[r][z-1]+
          T[r][(z==(ZTOTAL-1))*(z-1)+(z<(ZTOTAL-1))*(z+1)];
      else
        Tave=2*T[r+1][z]+2*T[r][z+1];
    else
      Tave=T[r-1][z]+
        2*T[r][z+1]+
        T[(r==(RTOTAL-1))*(r-1)+(r<(RTOTAL-1))*(r+1)][z];

c=ctab[(int)(10*FACTOR*T[r][z])];  /* J/K */
rho=rhocalc(T[r][z],field); /* Ohm cm */


/*
if ((outtime)&&(volts[r][z]>0)&&(z==0)&&((r==INN_LAYERS-1)||
(r==LAYERS-1)))
 {
   printf("T[%d][%d]=%f, field=%f, rho=%g\n",
         r,z,T[r][z],field,rho);
   fflush(stdout);
 }
 */
```

```
/* Compute Critical currents based on conductor paramaters */
if (T[r][z]<TC3)
  if (mtrl[r][z]==CONDUCTOR_OUT)
    Ic=COEFA2*T[r][z]*T[r][z]+COEFB2*T[r][z]+COEFC2;
/* Outer Coil */
  else
    Ic=COEFA1*T[r][z]*T[r][z]+COEFB1*T[r][z]+COEFC1;
/* Inner Coil  */
else
  Ic=0;


if (current>Ic)
  {
    volts[r][z]=
      (current−Ic)*rho*CIRCUMFERENCE/ CON_AREA;
    qnz=DT*volts[r][z]*current;

    /* printf(" ****CONDUCTOR NODE NORMAL****\n"); */


    if ((outtime)&&(volts[r][z]>0)&&(z==0)&&
        ((r==INN_LAYERS−1)||(r==LAYERS−1)))
      {
        printf("qsink=%f  qnz=%1.5e  volts=%1.5g  Ic=%3.3f",
               qsink,qnz,volts[r][z],Ic);
        printf("  current=%3.2f  local P=%1.4e  dT/dt=%2.3f [K/s]",
               current,qnz/DT,qnz/DT/c);
        if (mtrl[r][z]==CONDUCTOR_INN)
          printf("  Total P=%f  Inner Coil\n",
                 qnz/DT*TURNSpLAYER*INN_LAYERS);
        else
          printf("  Total P=%f  Outer Coil\n",
                 qnz/DT*TURNSpLAYER*OUT_LAYERS);
        fflush(stdout);
      }
    }
  else /* current < Ic */
    {
      qnz=0;
      volts[r][z]=0;
    }

  /*
  if ((outtime)&&(volts[r][z]>0))
    {
      printf(“ qnz=%g\tr=%d\tz=%d\n”,qnz,r,z); fflush(stdout);
    }
    */

  /* main calculation of new Temp */
  /*   T[r][z]=Tave/4+
      (double)(qcond+qnz+qtop-qbottom-qright[z]+qleft[z])/c; */
T[r][z]+=(double)(qcond+qnz+qtop-qbottom-qright[z]+qleft[z])/c;
/*
if (rm<invRright)
        rm=invRright;
if (rm<invRbottom)
        rm=invRbottom;
        */

/*
if ((outtime)&&(volts[r][z]>0)&&(z==0)&&
((r==INN_LAYERS-1)||(r==LAYERS-1)))
  {
    printf(“(%3.5f - %3.5f)\t=%f\n”,current, Ic,current-Ic);
    printf(“(%3.5f - %3.5f)* %g\t = %g\n”,
            current, Ic, rho, (current-Ic)*rho);
    printf(“(%3.5f - %3.5f)* %g * %f\t = %g\n”,
            current, Ic, rho, CIRCUMFERENCE,
            (current-Ic)*rho*CIRCUMFERENCE);
    printf(“(%3.5f - %3.5f)* %g * %f/ %f = %g\n”,
                current, Ic, rho, CIRCUMFERENCE, CON_AREA,
                (current-Ic)*rho*CIRCUMFERENCE/CON_AREA);
            printf("(%3.5f - %3.5f)* %g * %f* %f = %g\n",
                current, Ic, rho, CIRCUMFERENCE, 1/CON_AREA,
                (current-Ic)*rho*CIRCUMFERENCE*1/CON_AREA);
            printf("(%3.5f - %3.5f)* %g * %3.5f / %1.6f = %g\n",
                current, Ic, rho, CIRCUMFERENCE, CON_AREA, volts[r][z]);
            fflush(stdout);
          }
          */

        qtop=qbottom;

        if((T[r][z] > 500.0))    /* stop if temperature exceeds 500 K */
          {
            printf("time  [sec] \ttstep\tr\tz\tT[r][z]\t\t");
            printf("qnz\t\tq out\t\tc\t\tmtrl[r][z]\n");
            printf("%f\t%d\t%d\t%d\t%f\t%f\t%f\t%f\t",
                time*DT,time, r, z, T[r][z], qnz,qsink, c);
            if (mtrl[r][z]==CONDUCTOR_OUT)
                  printf("CONDUCTOR_OUT\n");
            if (mtrl[r][z]==CONDUCTOR_INN)
                  printf("CONDUCTOR_INN\n");
            r = RTOTAL;
            z = ZTOTAL;
            time = TIMESTEPS;
            stop=1;
          }      /* end... if (T> 500)  */
        } /* end z loop */
    } /* end r loop */

  if (outtime)
    {
      WriteTempData(outputT ,T);
      WriteVoltData(outputV, volts);
      /*
```

```
          printf("Timestep %d of %d at %f seconds of %f seconds\n",
                 time, TIMESTEPS, time*DT, TOTALTIME);
          fflush(stdout);
          */
        }
    } /* end time loop */
  /* printf("Max invR=%f\n",rm); fflush(stdout); */
}   /*** end TimeLoop ***/

/*********************************************************************/
/* function conductorcalc *********************************************/
/*********************************************************************/
float conductorcalc(double Tlocal, double Tneighbor, float ktab[TTOT][3])
{
  return ((Tlocal-Tneighbor)*CON_AREA/
          PI*RADIUS/(1/ktab[(int)(10*Tlocal)][0]+
                     1/ktab[(int)(10*Tneighbor)][0]));
}   /*** end conductorcalc ***/

/*********************************************************************/
/* function cryocooler ************************************************/
/*********************************************************************/
float cryocooler(double T)
{
  return ((T>TEMPOP)*DT*(P1*pow(T,2)+P2*(T)+P3));
}   /*** end cryocooler ***/

/*********************************************************************/
/* function invrightcalc **********************************************/
/*********************************************************************/
float invrightcalc(double T[RTOTAL][ZTOTAL],
                   float ktab[TTOT][3], int r, int z)
{
  return( 2*ktab[(int)(10*T[r][z])][0]* CIRCUMFERENCE* CON_W*
         ktab[(int)(5*(T[r+1][z]+T[r][z]))][1]*
         ktab[(int)(10*T[r+1][z])][0]/
         (CON_H*
          ktab[(int)(5*(T[r+1][z]+T[r][z]))][1]*
          ktab[(int)(10*T[r+1][z])][0]+
          6* INSULTHICKNESS*
          ktab[(int)(10*T[r][z])][0]*
          ktab[(int)(10*T[r+1][z])][0] +
          CON_H*
          ktab[(int)(10*T[r][z])][0]*
          ktab[(int)(5*(T[r+1][z]+T[r][z]))][1] +
          2* CONTACT_H*
          ktab[(int)(10*T[r][z])][0]*CIRCUMFERENCE*CON_W*
          ktab[(int)(5*(T[r+1][z]+T[r][z]))][1]*
          ktab[(int)(10*T[r+1][z])][0]));

}   /*** end invrightcalc ***/

/**********************************************************************/
/* function invbottomcalc ***************************************************/
/**********************************************************************/
float invbottomcalc(double T[RTOTAL][ZTOTAL],
                float ktab[TTOT][3], int r, int z)
{
    if (z<ZTOTAL-2)    /* within conductors */
      return(2*ktab[(int)(10*T[r][z])][0]*CIRCUMFERENCE*CON_H*
            ktab[(int)(5*(T[r][z]+T[r][z+1]))][1]*
            ktab[(int)(10*T[r][z+1])][0]/
            (CON_W*
             ktab[(int)(5*(T[r][z]+T[r][z+1]))][1]*
             ktab[(int)(10*T[r][z+1])][0]+
             4*INSULTHICKNESS*
             ktab[(int)(10*T[r][z])][0]*
             ktab[(int)(10*T[r][z+1])][0]+
             CON_W*
             ktab[(int)(10*T[r][z])][0]*
             ktab[(int)(5*(T[r][z]+T[r][z+1]))][1] +
             2*CONTACT_V*
```

```
                ktab[(int)(10*T[r][z])][0]*CIRCUMFERENCE*CON_H*
                ktab[(int)(5*(T[r][z]+T[r][z+1]))][1]*
                ktab[(int)(10*T[r][z+1])][0]));
}   /*** end invbottomcalc ***/

/**************************************************************************/
/* function CreateTables *************************************************/
/**************************************************************************/
void CreateTables(float ktab[TTOT][3], float ctab[TTOT*FACTOR], float field)
{

  int n, temp;
  printf("Begin CreateTables\n"); fflush(stdout);
  for(n = 0; n < 3; ++n)
    {
      for(temp = 0; temp < 10; ++temp)
        ktab[temp][n] = 0.0;
      for(temp = 10; temp < (TTOT); ++temp)
        {
          ktab[temp][n] = kcalc(((float)temp/10), n, field);
          /*  printf(" ktab[%d][%d]=%f\n",
                  temp,n,ktab[temp][n]);  fflush(stdout); */
        }
    }

      for(temp = 0; temp < 10; ++temp)
        ctab[temp] = 0.0;
      for(temp = 10; temp < (TTOT*FACTOR); ++temp)
        {
          ctab[temp] = ccalc(((float)temp/10/FACTOR), 0);
          /*  printf(" ctab[%d]=%f\n",
                  temp,ctab[temp]);  fflush(stdout); */
        }


 printf(" CreateTables(ktab, ctab, field); --Done--\n"); fflush(stdout);
```

```
}   /*** end CreateTables ***/

/**********************************************************************/
/* function InitializeFile ********************************************/
/**********************************************************************/
void InitializeFile(char name[])
{
  FILE *ofp;

  printf("Begin InitializeFile\n"); fflush(stdout);
  printf(" name[] is: %s \n",name); fflush(stdout);
  ofp = fopen(name, "w");
  fprintf(ofp, "");
  fclose(ofp);
  printf(" InitializeFile(%s); --Done--\n",name); fflush(stdout);
}   /*** end InitializeFile ***/

/**********************************************************************/
/* function InitializeMaterials ***************************************/
/**********************************************************************/
void InitializeMaterials(int mtrl[RTOTAL][ZTOTAL])
{
  int r,z;

  printf("Begin InitializMaterials\n"); fflush(stdout);

  for (z=0; z<ZTOTAL; z++)
    for (r=0; r<INN_LAYERS; r++)
      mtrl[r][z]=CONDUCTOR_INN;
  for (z=0; z<ZTOTAL; z++)
    for (r=INN_LAYERS; r<LAYERS; r++)
      mtrl[r][z]=CONDUCTOR_OUT;

  printf(" mtrl[][] values \n"); fflush(stdout);
  for (z=0; z<ZTOTAL; ++z)
    {
      for (r=0; r<RTOTAL; ++r)
        printf(" %2d",mtrl[r][z]); fflush(stdout);
      printf("\t end row z=%d\n",z); fflush(stdout);
    }

  /* end of printing materials section */
  printf(" InitializeMaterials  -- Done --\n"); fflush(stdout);
}    /*** end InitializeMaterials ***/



/***********************************************************************/
/* function rho0calc ***************************************************/
/***********************************************************************/

/* zero field resistivity calculation for silver */
float rho0calc(float T)
{

float    m0 = .069136;
float    m1 = -.006714;
float    m2 = .00019844;
float    m3 = -9.728e-7;
float   rho0;

  if(T>=70.0)
    {
    m0 = -.34145;
    m1 = .0094905;
    m2 = -1.9905e-5;
    m3 = 3.2803e-8;
    }

  if(T<300.0)
    rho0 =     (m0 + (m1 * T) +
               (m2 * T * T) +
               (m3 * T * T * T)) * 1e-6;
  else rho0 = ((2.6775 * log10(T/300.0)) + 1.6) * 1e−6;

  return (rho0);
}      /*** end rho0calc ***/

/**********************************************************************/
/* function rhocalc ***************************************************/
/**********************************************************************/

/*  nonzero field resistivity calculation for silver */
float rhocalc(float T,float b)
{

  float m4, m5, rho0, rho;
  float rho0calc(float T);

  rho0 = rho0calc(T);

  if((b * (1.43338e−6)/rho0) < 200.0)
    {
      m4 = .00014736;
      m5 = 1.5838;
    }
  else
    {
      m4 = .117;
      m5 = .76403;
    }

  rho = (rho0 * m4 * pow((b * (1.43338e−6)/rho0), m5)) + rho0;
  return rho;
}   /*** end rhocalc ***/



/**********************************************************************/
```

```
/* function kcalc **********************************************************/
/***************************************************************************/

/* thermal conductivity for all materials */
float kcalc(float T, int mtrl, float b)
{
  float k, k0, rho, rho0;
  float rhocalc(float Temp, float field);
  float rho0calc(float Temp);

  /* printf(" Begin kcalc  CONDUCTOR_OUT=%d INS1=%d INS2=%d CF=%d\n",
     CONDUCTOR_OUT, INSULATOR1, INSULATOR2, COILFORM); fflush(stdout);
     printf("  Input values\n"); fflush(stdout);
     printf("  T=%f\tmtrl=%d\tb=%f\n",T,mtrl,b); fflush(stdout); */

  if (mtrl == 0)  /* CONDUCTOR_OUT */
   {
    rho = rhocalc(T,b);
    rho0 = rho0calc(T);

    if(T<300)
      k0 = 4.0 + (3.2 * T * T * pow(2.71828, (4.0 − T)/4.5));
    else k0 = 4.0;

    k = SILVERFRACTION * (1 + (0.05 * b)) * (rho0 / rho) * k0;
   }

/* mtrl = 0  CONDUCTOR_OUT
   mtrl = 1  INSULATOR 1 or 2
   mtrl = 2  COILFORM  */

  else if(mtrl == 1)
    k = (0.0015335 + (.0000227 * T));
  else if(mtrl == 2)
    /* modify function to calculate for coilform */
    if (T<300)
```

```
      k = 4.0 + (3.2 * T * T * pow(2.71828, (4.0 - T)/4.5));
    else
      k=4;



  /* printf(“  Final values\n”); fflush(stdout);
     printf(“  T=%f\tmtrl=%d\tb=%f\n”,T,mtrl,b); fflush(stdout);
     printf(“  k=%f\n”,k); fflush(stdout);
     printf(“   return k\n”); fflush(stdout); */
  return (k);
}    /*** end kcalc ***/



/*************************************************************************/
/* function ccalc ********************************************************/
/*************************************************************************/

/* heat capacity for all materials **/
float ccalc(float T, int mtrl)
{
  float c,C=0;

  /*  Units are J/cm^3K in interpolation tables */
  /*  Add the relavant volumes to obtain total c*/

  /*
  printf(“Begin ccalc\n”);  fflush(stdout);
  printf(“ Input Values T=%f\tmtrl=%d\n”,T,mtrl); fflush(stdout);
  */

/* mtrl = 0  CONDUCTOR node
   mtrl = 1  COILFORM  (straight Silver)
   */
  if(mtrl == 0)
   {
     c =((1.0 + SILVERFRACTION)/2)*
       pow((pow((17.75e-6 * T * T * T),
                 -0.85)+ 0.47081), -1.1765);
     C +=c*CIRCUMFERENCE*CON_AREA;
     /*
     c =(-.016 + (.002074 * T) + (1.7738e-4 * T * T));
     C+=c*INSULTHICKNESS*
     (4*INSULTHICKNESS+2*CON_H+3*CON_W)*CIRCUMFERENCE;
     */
   }
  else if((mtrl == 1))
   {
     c =pow((pow((17.75e-6 * T * T * T), -0.85)+ 0.47081), -1.1765);
     C=0;
   }

  /* printf(“ Final Values T=%f\tmtrl=%d\tc=%f\n”,T,mtrl,C);
    fflush(stdout);
    printf(“  return c\n”); fflush(stdout); */

  return C;
}   /***  end ccalc ***/
```

Appendix B

Matlab Post-Processing Code

```
% process.m  - simulation processing
%  uses data output from prog.c to graph simulation results
%  should be run from the directory containing the siminfo.m,
%     temp.dat, and volt.dat files produced by the program

clear;

siminfo

% The first TURNSpLAYER lines are the first set.
% Each new sampling simply adds new lines to the old.
% The dimensions of the matrix is therefore rows,columns
%      with set 1 having (1:TURNSpLAYER, 1:LAYERS).
%      Each additional set will add successive TURNSpLAYER to
%      these numbers.

disp('What do you want to graph them?')
disp('   1. Temperature data')
disp('   2. Voltage data')
choice=input(' Choice? [2] ','s');

if length(choice)==0
  volts
```

```
elseif choice(1)=='1'
   temps
elseif choice(1)=='2'
   volts
end;
```

```
% volts.m

% This file is called by process.m as a way of creating
% graphs of voltage data from the C program.
%
% It assumes that the following variables have allready been set:



volt=load(Voltfile);
[c,d]=size(volt);

sets=floor(TIMESTEPS/SAMPLERATE);
sets=floor(c/(ZTOTAL));

clear choice k
disp(['The voltage data file contains 1:' num2str(sets) ' samples'])
disp('How do you want to graph them?')
disp('   1. Plot voltages traces for inner and outer coils.')

choice=input(' Choice? [1] ','s');

vtraces
```

```
% temps.m

% This file is called by process.m as a way of creating
% 3D temperature profiles of data output by the C program.
%
```

```
% It assumes that the following variables have allready been set by siminfo:
%

temp=load(Tempfile);
[a,b]=size(temp);

sets=floor(TIMESTEPS/SAMPLERATE);
sets=floor(a/(ZTOTAL));

clear choice k;
disp(['The Temperature data file contains ' num2str(sets) ' samples'])
disp('How do you want to graph them?')
disp('   1. Surface plots of temperature data')
disp('   2. Time vs Temp for specific nodes')
disp('   3. Plot statistical teperature data')
choice=input(' Choice? [1] ','s');

if length(choice)==0
   k=1;
elseif choice(1)=='1'
   k=1;
elseif choice(1)=='2'
   k=2;
elseif choice(1)=='3'
   k=3;
else
        k=1;
end;

switch k
       case 1,
            temp3d
       case 2,
            temp2d
       case 3,
            temp1d
```

```
    otherwise,
        error('Selection error')
end;
```

```
% vtraces.m

% This file is called by volts.m as a way of creating
% graphs of data output by the C program.
%
% It assumes that the following variables have allready been set:
%   sets
%   [c,d] = size(volt)
%
%      RefFile  (a string)
%      Tempfile (a string)
%      Voltfile (a string)
%      LAYERS
%      TURNSpLAYER
%      TOTALTIME
%      SAMPLERATE
%      ZTOTAL
%      RTOTAL
%      DT
%      TIMESTEPS
%      INN_LAYERS
%      OUT_LAYERS

% voltage begins at the upper left corner, (0,0) and moves first down,
%   then up through the coil.  The INN_LAYER must be even
sets=sets-1;
for i=1:sets
      voltage1(i)=sum(sum(volt(i*ZTOTAL:(i+1)*ZTOTAL-1,1:INN_LAYERS)));
      voltage2(i)=sum(sum(volt(i*ZTOTAL:(i+1)*ZTOTAL-1,INN_LAYERS+1:LAYERS)));
end;
```

```
time=DT*SAMPLERATE*(1:sets);
trnum=str2num(RefFile(length(RefFile)));
peakvr1=[0 0 0 .01 .031 .0507];
peakvr2=[.0156 .0732 .3516 .049 .1758 .4395];
ptimes=[3.84 2.82 3.28 3.06 2.94 3.0];
peak1=peakvr1(trnum);
peak2=peakvr2(trnum);
ptime=ptimes(trnum);
mark='ko';

figure(3)
subplot(2,1,1)
plot(time,voltage1,'b-',ptime,peak1,mark)
set(gca,'Fontsize',14);
title(['Simulated Voltage Traces in ' RefFile],'Fontsize',16)
A=axis;
if A(4)<=peak1
        A(4)=peak1*1.1;
end;
axis(A);
legend('Simulated trace', 'Experimental peak',2)

ylabel('Inner coil [V]','Fontsize',14)
subplot(2,1,2)
plot(time,voltage2,'b-',ptime,peak2,mark)
set(gca,'Fontsize',14);
A=axis;
if A(4)<=peak2
        A(4)=peak2*1.1;
end;
axis(A);
ylabel('Outer coil [V]','Fontsize',14)
xlabel('Time [s]','Fontsize',14)
legend('Simulated trace', 'Experimental peak',2)
```

```
disp(RefFile)
disp(['   Outer Coil Peak ' num2str(max(voltage2))])
disp(['   Inner Coil Peak ' num2str(max(voltage1))])
```

```
% temp3d.m

% This file is called by temps.m as a way of creating
% 3D temperature profiles of data output by the C program.
%
% It assumes that the following variables have allready been set:
%    temp      data matrix loaded from 'tempfile'
%    sets      number of sample sets taken
%    a,b       size of array

clear choice, k;
disp(['The Temperature data file contains 1:' num2str(sets) ' samples'])
disp('How do you want to graph them?')
disp('   1. Plot all surfaces in the sets')
disp('   2. Choose a range of sets')
disp('   3. Choose specific sets')
disp('    4. Plot nodes near cryocooler')

choice=input(' Choice? [1] ','s');
if length(choice)==0
  k=1;
  range=1:sets;
elseif choice(1)=='1'
  k=1;
  range=1:sets;
elseif choice(1)=='2'
  k=2;
  range=input('Enter range for sets a:b ');
elseif choice(1)=='3'
  k=3;
  range=input('Enter sets as a space or comma separated list ');
```

```
elseif choice(1)=='4'
  k=4;
  noder=input('Plot r nodes from 0 to ');
  nodez=input('Plot z nodes from 0 to ');
  range=input('Enter set range as [start:step:finish] ');
end;



if k==4
   i=range(length(range));
   figure(1)
   grid=temp(((i-1)*ZTOTAL+1):((i-1)*ZTOTAL+1+nodez),1:1+noder);
   surf(grid');
   AZ=axis;
   close(1);
  for i=range
   figure(100+i);
   grid=temp(((i-1)*ZTOTAL+1):((i-1)*ZTOTAL+1+nodez),1:1+noder);
   surf(grid');
   xlabel('z node number')
   ylabel('r node number')
   zlabel('Temp [K]')
   title(['Temperature profile at ' num2str(i*SAMPLERATE*DT) ' sec'])
   %axis([1, nodez+1, 1, noder+1, AZ(5:6)])
  end;
else
 for i=range
   figure(100+i);
   surf((temp(((i-1)*ZTOTAL+1):((i)*ZTOTAL),:))');
   xlabel('z node number')
   ylabel('r node number')
   zlabel('Temp [K]')
   title(['Temperature profile at ' num2str(i*SAMPLERATE*DT) ' sec'])
   AZ=axis;
      %view([144,17])
   %axis([1 62 1 48 20 110])
  cnd;
end;
```

```
% temp2d.m

% This file is called by temps.m as a way of creating
% 2D temperature profiles of data output by the C program.
%
% It assumes that the following variables have allready been set:
%     temp       data matrix loaded from 'tempfile'
%     sets       number of sample sets taken
%     a,b        size of array

clear choice, k;
disp(['The Temperature data file contains ' num2str(sets) ' samples'])
disp('Which nodes do you want to graph?')
disp(' 1.  Plot 4 equally spaced verticle exterior nodes')
disp(' 2.  Plot 4 equally spaced horizontal nodes on centerline')
disp(' 3.  Pick specific node numbers')
disp(' 4.  Plot z=0 nodes for r=0,1,2,3,4,5,6,7')

range=1:sets;
choice=input(' Choice? [1] ','s');
if (length(choice)==0)
   k=1;
elseif choice(1)=='1'
   k=1;
elseif choice(1)=='2'
   k=2;
elseif choice(1)=='3'
   k=3;
   disp(' Using the Matlab convention that numbers start from 1,')
clseif choice(1)=='4'
   k=4;
end;
```

```
switch (k)
    case 1,
    zs=floor(1:TURNSpLAYER/4:TURNSpLAYER);
    nodes=[LAYERS, zs(1); LAYERS, zs(2); LAYERS, zs(3); ...
            LAYERS, zs(4)];
    nodesb=[nodes(:,2),nodes(:,1)];
    case 2
    rs=floor(1:LAYERS/4:LAYERS);
    nodes=floor([TURNSpLAYER/2, rs(1); TURNSpLAYER/2, rs(2); ...
                  TURNSpLAYER/2, rs(3); TURNSpLAYER/2, rs(4)]);
    nodesb=[nodes(:,2),nodes(:,1)];
    case 3,
    disp('enter sets as a 2 column matrix [r(1),z(1);r(2),z(2); ...')
    nodesb=input('Type square brackets manually ')
    nodes=[nodesb(:,2),nodesb(:,1)];
    case 4,
    nodes=[1 0; 2 0; 3 0; 4 0; 5 0; 6 0; 7 0];
    nodesb=[nodes(:,2),nodes(:,1)];
end;

[alpha,beta]=size(nodes);
if beta~=2
    error('Incorrect node arguments')
end;

    for a=1:alpha
        for i=1:sets
    str2eval=['node' num2str(a) '(' num2str(i) ...
         ')=temp(((' num2str(i-1) ')*ZTOTAL+' ...
         num2str(nodes(a,1)) '),' num2str(nodes(a,2)+1) ');'];
            eval(str2eval);
        end;
    end;

%for i=1:sets
```

```
%       node0(i)=temp(((i-1)*ZTOTAL+1),1);
%       node1(i)=temp(((i-1)*ZTOTAL+1),2);
%       node2(i)=temp(((i-1)*ZTOTAL+1),3);
%       node3(i)=temp(((i-1)*ZTOTAL+1),4);
%       node4(i)=temp(((i-1)*ZTOTAL+1),5);
%       node5(i)=temp(((i-1)*ZTOTAL+1),6);
%       node6(i)=temp(((i-1)*ZTOTAL+1),7);
%       node7(i)=temp(((i-1)*ZTOTAL+1),8);
%   end;
colorcode=['m';'c';'r';'g';'b';'k';'y'];
linecode=['- ';': ';'-.';'--'];

figure(1);
time=0:SAMPLERATE*DT:(sets-1)*DT*SAMPLERATE;
[alpha,beta]=size(nodes);
for i=1:alpha
        linetype=[colorcode(mod(i,7)+1,:),linecode(mod(i,4)+1,:)];
        nodenum=eval(['node' num2str(i)]);
        plot(time,nodenum,linetype);
        set(gca,'Fontsize',14);
        hold on
end;
xlabel('time [s]','Fontsize',14)
ylabel('temperature [K]','Fontsize',14)
title([RefFile],'Fontsize',14)
hold off
legstr='';
for i=1:alpha
        addstr=['r=' num2str(nodesb(i,1)) ', z=' num2str(nodesb(i,2))];
        newstr=blanks(13);
        newstr(1:length(addstr))=addstr;
        legstr=[legstr; newstr];
end;
legend(legstr,0);
set(gca,'Fontsize',14);
```